Humboldt Universität zu Berlin
School of Business and Economics
Institute for Statistics and Econometrics
Ladislaus von Bortkiewicz Chair of Statistics

Automating Exploratory Data Analysis for use in Predictive Modeling (Classification)

Master's Thesis

by

Alena Churakova

Advisor

Dr. Sigbert Klinke

Examiners

Prof. Dr. Wolfgang Karl Härdle

Prof. Dr. Marlene Müller

March 11, 2017

Acknowledgment

I would first like to thank Dr. Annina Neumann, then Head of the Data Science team of SHS VIVEON AG, who originated the project and granted me this incredible opportunity. Annina provided me insightful guidance, challenged my ideas and helped me to keep focused on the goals. I am deeply thankful to Dr. Alexander Kaul who tirelessly supported the project on its later stages. As Head of the Data Science, he actively involved the team members into the development, which doubtlessly improved the result. I am also thankful to each one of the members of the Data Science team of SHS VIVEON AG who generously contributed their time and effort.

I would like to express my gratitude to Dr. Sigbert Klinke, my thesis advisor, who embraced the idea of this practice-oriented work and offered me encouraging guidance throughout the whole journey. His suggestions and feedback every time brought me further and I believe are helping me to become a better statistician and a better R programmer.

My thanks also go to R contributors and a vital online community of R users who made the creation of a software tool described in this thesis possible.

I am grateful to my friends and family who offer me constant support and help. To Katja whom I met as my math tutor many years ago and who become my friend and inspiration. She coached me on scientific writing and her feedback on this thesis was eye-opening. To my dearest friend Jeremy who helped me to sharpen the arguments and the language of this work. To my partner for his patience and endless support. To my parents who always encourage me and without whom I would not become who I am. To my grandfather, deda Vasja, whose passion for life, physics and mathematics shaped my world.

Abstract

A data analysis tool was developed using R in order to quickly and efficiently gain an initial understanding of the structure and predictive power of a dataset. This tool was developed for SHS VIVEON AG, a business and IT solution provider for customer management, and is meant to enable the firm's Data Science team to perform initial exploratory data analysis of a client's dataset. The two key drivers in the design of the software were versatility in the functions (visualization, clustering, classification, etc.) and an intuitive interface. This work will do the following: outline the needs of the firm's Data Science team when faced with a new dataset, review the statistical and analytic methods for engaging with data in a business setting, recount the steps taken to create the tool in question (including the full considerations of SHS VIVEON AG and the results of usability testing), and take the reader through a step-by-step mock analysis of a non-proprietary dataset. Overall, the current work shows the background, design, and implementation of this software, explaining both its technical and interactive features. Finally, it points to ways that users can modify and improve upon the software if other requirements are identified in the future.

Keywords: predictive modeling, exploratory data analysis, software, classification, R

Contents

List of Figures

List of Tables

Abbreviations

AUC	area under curve
CLARA	Clustering LARge Applications
CV	cross-validation
DDP	Data Discovery Platform
EDA	exploratory data analysis
FN	false negative
FP	false positive
IQR	interquartile range
k-NN	k-nearest-neighbor
MADM	median absolute deviation from the median
NPV	negative predictive value
PAM	partitioning around medoids
PC	principal component
PCA	principal component analysis
PPV	positive predictive value
ROC	receiver operative characteristic
TN	true negative
TP	true positive

1 Introduction

This thesis describes a statistical software tool that was developed for the Data Science team of SHS VIVEON AG, a business and IT solution provider for customer management. Developed to address the requirements of the two main stakeholders, i.e. the firm and the end user (analyst), the software is an interactive application that allows users to perform a thorough initial exploratory analysis of data without the need for scripting.

Since the seminal work by Tukey (1977), exploratory data analysis (EDA) has often been understood as an attitude and approach to flexible investigation of data without assumptions about the data generating process (Kotz et al. 2006, 11:2152). The amount and complexity of data produced by businesses is constantly growing. Likewise, new techniques for data analysis are being rapidly developed to keep pace with this growing complexity. The advances in algorithmic data processing do not undermine the importance of understanding the data at hand, but continue to profit from it.

This exploratory data analysis tool provides assistance in answering questions about the appropriateness of clients' data for addressing specific problems, e.g. prediction. It also provides estimates about the time, effort and resources required for a project. The scale and possible consequences of such decisions require a tool that is reliable and capable of providing trustworthy results. Success and profitability may depend on it. At the same time, an end user of this tool should perceive benefits of use for themselves. The tool should be pleasant and intuitive to use, and offer an appropriate scope of functions and features.

Striking a balance between functionality and simplicity is one of the core challenges of the digital product development. Excessive complexity and a steep learning curve may discourage users from investing time into learning, while the lack of highly desired features would force users to turn to additional tools to complete an analysis, thereby reducing the appeal of the software.

In a typical scenario, SHS VIVEON's Data Science team receives tabular data from its client, from varying industries, e.g. telecommunication, automotive. The historic data is used to help the client to increase understanding of its business, and to make predictions about events of interest.

The company considered automating the repetitive task of initial exploratory analysis in order to make a more efficient use of its time and resources and also to make the process a more pleasant experience for its analysts. Therefore, it was imperative that the software be able to:

- quickly provide a first impression of the information in the data, including predictive performance;
- be manageable via a graphical interface; and

- securely operate on an internal server with a Linux operating system (where the client data is stored).

The software tool is limited to a qualitative outcome variable, i.e. to classification, because of the high frequency of such requests from clients. Both binary and multiclass problems are able to be addressed by the software. The back end is powered by R, an open-source programming language for statistical computing and graphics. The front end is implemented as a web-based interface.

This work consists of three main parts. Section 2 contains the theoretical foundation regarding the methods used in the software tool. It is meant to be a basic reference for the functions used in the tool rather than an exhaustive explanation of all data exploration techniques. The structure, functionality, and technical implementation of the data analysis tool are explained in Section 3, which also includes information on the process and results of user testing. Finally, Section 4 presents an application of the developed tool using real-world data. Appendix A contains screenshots of all software tabs and Appendix B lists files on the enclosed CD.

2 Theory

This section provides background for descriptive statistics and statistical learning theory that supports methods implemented in the software tool. The explanatory focus of the work only takes into account sample statistics and no inferential matters are examined. In other words, the emphasis lies on flexible analysis of the sample without assuming any probabilistic model.

2.1 Data preparation

2.1.1 Data matrix and variable types

Tabular data consists of a collection of n observations with p measurements, building an $n \times p$ data matrix. Rows can be referred to as instances, cases, objects, units, records or observations ($i = 1, \ldots, n$), while columns – as variables or fields ($j = 1, \ldots, p$).

Measurements in each column vary by type, a characteristic crucial for the applicability of analytic techniques. A high level distinction is made between variables that are measured on a numerical scale, those that have an order, but no scale, and those that have neither scale no ordering. The first kind can be characterized by a wide range of statistics as mean, sum, difference, etc, while the last kind accepts the narrowest range of operations. This work treats variables either as quantitative (numeric) or qualitative (categorical, nominal). Various classifications of measurement scales and variable types are presented in Agresti (2002, 2–4) and Bortz and Schuster (2010, 12–23).

2.1.2 Constant and almost constant variables

Constant variables (also called unary or zero variance variables) contain identical values for all observations. Qualifying a variable as almost constant (near-zero variance) depends on the definition. Kuhn and Johnson (2013, 44–45) suggest two rules of thumb to identify near-zero variance fields: a low fraction of unique values and a large frequency ratio of the most prevalent to the second most prevalent values. A combination of these rules aims at preventing false categorization of more or less evenly distributed variables with low granularity (Kuhn 2008, 4).

2.1.3 Missing values

Missing values are entries of a data matrix that were not observed. Although there is no single convention on how to mark missing values, special symbols (e.g. `NA` in R) and numbers outside of plausible values range (e.g. negative for age) are used for coding.

Ways of dealing with missing data include the deletion of variables, the deletion of cases, imputation (see details e.g. in Enders 2010; Little and Rubin 2002) and the use of statistical methods that can deal with the issue of missing data internally. A column with all missing values has no explanatory power, while a variable with mostly missing values may require careful inspection. Their imputation tends to be risky and such variables are candidates for dropping from the analysis. Sometimes, a highly structured pattern of missingness (e.g. observations missing in one variable has values in another) may signal unfortunate coding of variables and such columns can be merged into one with appropriate coding (Pearson 2011, 98–99). In any case, a pattern of missing values may contain important information about the dataset.

2.1.4 Extreme values

The definition of an outlier as "an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data" (Barnett and Lewis 1994, 7) seems to be adopted and referenced by many researchers. Importantly, this raises the question of whether an observation belongs to a population. The name "extreme values" has the intention of departing from inference questions and concentrating on the exploration of the sample data.

The probabilistic approach to univariate outliers detection is based on the properties of the normal distribution. The probability of being beyond a specific number of standard deviations from the mean is fixed. In practice, population moments are unknown and they are routinely estimated (Wilcox 2010, 29–32). A univariate decision rule to detect extreme values is presented in Equation (1). A value x_{ij} from a data matrix described in Section 2.1.1 is identified as extreme if it is located further than t sample measures of spread $\hat{s}_j$ from a measure of location $\hat{c}_j$ of that variable (Pearson 2011, 310). Definitions of sample measures can be found in Upton and Cook (2002).

$$x_{ij} = \begin{cases} \text{extreme value} & \text{if } |x_{ij} - \hat{c}_j| / \hat{s}_j > t \\ \text{nonextreme value} & \text{otherwise} \end{cases} \tag{1}$$

where: $\hat{c}_j$ – sample mean or median
$\hat{s}_j$ – sample standard deviation or median absolute deviation from the median (MADM)
t – positive number

Two versions of the rule differ in their resistance to contaminated (e.g. erroneous) values. Sample breakdown point defines resistance of an estimator as the smallest fraction of the data for which arbitrary changes would distort the resulting estimate (Wilcox 2010, 16).

An alternative approach to univariate extreme values detection is based on the interquartile range (IQR) where values can be visually inspected on a box plot (Wilcox 2010, 34–35). Further rules can be found in Kotz et al. (2006, 9:5886–5887). Altogether, such procedures consider one variable at a time. "Anomalous" observations in more dimensions can be of interest (e.g. identified by relplot for bivariate data, see Wilcox 2010), but are outside of the scope of this work.

2.1.5 Data transformation

Transformation is a data preparation step that allow the subsequent analysis to be more accurate and revealing (Kotz et al. 2006, 14:8703). Hoaglin et al. (1983) considered re-expression one of the main ingredients of EDA, because it helps to emphasize important structures that are not evident at the original scale. At the same time, the structures detected via transformation can be artifacts of a sample and the supervision of a domain expert is needed (Hand et al. 2001, 38–39).

The first group of transformation techniques for numeric variables, e.g. logarithm or raising to a power, aims to achieve better statistical properties of the variable in question. Sometimes this happens at the cost of interpretability (Yu 2010, 11–12). Additionally, some transformations change the variable type, e.g. discretization/binning of numeric variables. The second type of data transformations aims to standardize variables to a consistent range for the purpose of comparability, e.g. min-max transformation results in range [0,1]), decimal scaling in [−1,1], centering by mean and dividing by standard deviation (z-score standardization), or centering by median and dividing by IQR (Flach 2012, 314; Larose and Larose 2015, 30–32).

2.1.6 Proximity measures

Clustering techniques and some classifiers are founded on the notion of similarity and dissimilarity. While these terms indeed measure the opposite, they seek to quantify proximity. Additionally, the term "distance" is often used as a synonym to dissimilarity

for quantitative data. This section discusses measures between units and between groups of units.

Inter-individual proximity measures depend on variable type (quantitative, nominal, etc.) and in part on the type of unit (sample or population) (Kotz et al. 2006, 12:7730–7737). Euclidean and Manhattan distances (Equations (2) and (3)) are only applicable to numeric variables while generalized Gower's dissimilarity (Equation (4)) accepts both qualitative and quantitative inputs. Further measures are described in Everitt et al. (2011, 46–56) or Flach (2012, 231–237).

Proximities for datasets with numeric or mixed data types are representable as a $n \times n$ dissimilarity matrix $\mathbf{D}$ with elements $d_{ii'}$, dissimilarities between ith and i'th object. Here an later in this work, bold uppercase characters denote matrices and lowercase bold characters – vectors. Proximity measure for a pair of observations includes dissimilarities in each of the p attributes and w_j, weights of j-th attribute.

$$\text{Euclidean distance} \quad d_{ii'} = \left[\sum_{j=1}^{p} w_j^2 (x_{ij} - x_{i'j})^2 \right]^{1/2} \tag{2}$$

$$\text{Manhattan distance} \quad d_{ii'} = \sum_{j=1}^{p} w_j |x_{ij} - x_{i'j}| \tag{3}$$

$$\text{Generalized Gower's dissimilarity} \quad d_{ii'} = \frac{\sum_{j=1}^{p} w_j \delta_{ii'}^{(j)} d_{ii'}^{(j)}}{\sum_{j=1}^{p} w_j \delta_{ii'}^{(j)}} \tag{4}$$

where:
- w_j – nonnegative weight of j-th variable
- $x_{ij}, x_{i'j}$ – j-th variable value for units i and i'
- $\delta_{ii'}^{(j)}$ – 0 if at least one variable is missing or asymmetric binary variable have 0-0 match, 1 otherwise
- $d_{ii'}^{(j)}$ – contribution of j-th variable to dissimilarity between i and i'

When variable j is nominal, $d_{ii'}^{(j)}$ is 0 if two values are equal and 1 if they are not. For numeric variables $d_{ii'}^{(j)}$ is a Manhattan distance after min-max transformation: $d_{ii'}^{(j)} = |x_{ij} - x_{i'j}|/R_j$, where R_j is the range of variable j. Ordinal values are substituted by ranks and treated as numeric (Kaufman and Rousseeuw 1990, 32–37; Maechler et al. 2016; Everitt 1992, 49–56).

Distance calculations according to Equations (2) and (3) consider only those variables for which values in both observations are present. If both units have no nonmissing values in common, the resulting dissimilarity cannot be defined. Possible solutions include the removal of units or the imputation of distances or values in the original data matrix (Kaufman and Rousseeuw 1990, 15; Hastie et al. 2009, 507).

Inter-group proximity measure is a dissimilarity $d(G, H)$ between groups G and H computed using pairwise proximities $d_{ii'}$ when i is in G and i' in H.

$$\text{Nearest-neighbour distance / Single linkage} \quad d(G,H) = \min_{i \in G, i' \in H} d_{ii'} \tag{5}$$

$$\text{Furthest-neighbour distance / Complete linkage} \quad d(G,H) = \max_{i \in G, i' \in H} d_{ii'} \tag{6}$$

$$\text{Average linkage} \quad d(G,H) = \frac{1}{n_G n_H} \sum_{i \in G} \sum_{i' \in H} d_{ii'} \tag{7}$$

where: n_G, n_H – number of observations in groups G and H respectively

2.2 Measures of association

Measures of association quantify the statistical strength of the relationship between a pair of variables. Measures are often scaled to either [0,1] or [−1,1] which facilitates comparison. The choice of association measures depends on variable type and, in particular, their possible mathematical characterizations.

Between numeric variables

Sample Pearson correlation between two continuous variables is probably the most widely used measure of association. Conceptually, it quantifies the reduction in error of prediction. It compares values of a variable with and without knowledge of the other variable. Spearman rank correlation coefficient (Spearman's ρ) substitutes variable values with ranks and it measures the monotone relationship rather than just the linear one addressed by the Pearson correlation. Another rank-based coefficient, Kendall's τ_b explicitly accounts for ties. Mathematical formulation and interpretation of coefficients can be found in Pearson (2011, 450–469, 490–491), Härdle et al. (2015, 438–450).

While some association measures are established for continuous variables, under certain conditions they are applicable to variables of other types. Pearson product-moment correlation coefficient is useful for integer-valued data for which distance-related quantities (e.g. mean) are well defined. Spearman rank correlation coefficient may also be useful for such data, however, it suffers from ties resulting from repetitive values (Pearson 2011, 491–493).

Between categorical variables

Distance-based quantities are not well defined for categorical variables and, therefore, a Pearson correlation coefficient is not applicable. Rank based measures may benefit ordinal variables but not nominal ones.

A contingency table is a usual way to summarize two nominal variables and a χ^2 statistic indicates whether there is a statistical relationship. However, this statistic depends on sample size and table dimensions (number of rows and columns). Many contingency table based measures adjust the χ^2 statistic to the sample size, the dimensions of the table, or both. These include ϕ (ϕ^2), contingency coefficient, and Cramér's V (definitions can be found e.g. in Everitt 1992, 54–55). While all metrics are bound by zero from below (no relationship), the first two do not have a desirable clear upper bound. The dependence of the upper bound on table dimensions is further adjusted in Cramér's V. For this reason, Cramér's V is a more universal measure (Bortz and Lienert 2008, 271–275).

Another group of measures is based on reduction of error in prediction. They incorporate predictive ability into meaning of association. A range of metrics was suggested by Goodman and Kruskal λ, many of which are asymmetric (see details and examples in Everitt 1992, 55–59 or Kotz et al. 2006, 245–251).

Between numeric and categorical variables

When measuring an association between variables of different types, a metric for the less flexible type is applicable, e.g. ordinal and nominal variables should be treated as nominal which results in loss of information of order (Pearson 2011, 491–494). The focus of the developed software on classification requires measuring association between a numeric variable and a categorical target variable. Therefore, effect sizes for subgroup analysis are considered.

One of the effect sizes based on proportion of variance explained, (partial) η^2, measures strength of association between a dependent quantitative and independent qualitative variables in a sample. In case of only one factor, partial η^2 and η^2 are both equal to a proportion of the total variance in the dependent variable attributable to the factor. This sample effect size is calculated as a ratio: a sum of squared differences between the values of the numeric variable and the group means in the numerator and the total sum of squared differences between these values and their mean, ignoring the groups, in the denominator (Tabachnick and Fidell 2001, 36–37, 52–53).

2.3 Data visualization

Visual exploration of data is critical to it being understood. Different graphs are used to present summary statistics, display distributions, show relationship between variables, facilitate choice of model parameters, detect groups, etc. Moreover, the choice of graphical form depends on the type of the depicted data (Clarke et al. 2009, 531–534).

Plots designed for numeric variables depict several features such as asymmetry, multimodality, heaping, extreme or impossible values. Displays of categorical variables may reveal uneven distributions, repeated categories, and unexpected patterns (Un-

win 2015, 27–29, 53–56).

Bar charts depict absolute or relative frequencies for each level of a categorical variable. They are also used to plot some quantitative measures (e.g. means). In the latter case, values can be complemented by error bars. A clustered (grouped) bar chart (Figure 1a) visualizes a contingency table of two categorical variables. **Mosaic plots** (Figure 1b) are another way of presenting contingency tables of two or more categorical variables: the space of the graph is subdivided into rectangles proportional to the counts of the multivariate combinations of categories (Unwin 2015, 131–137).

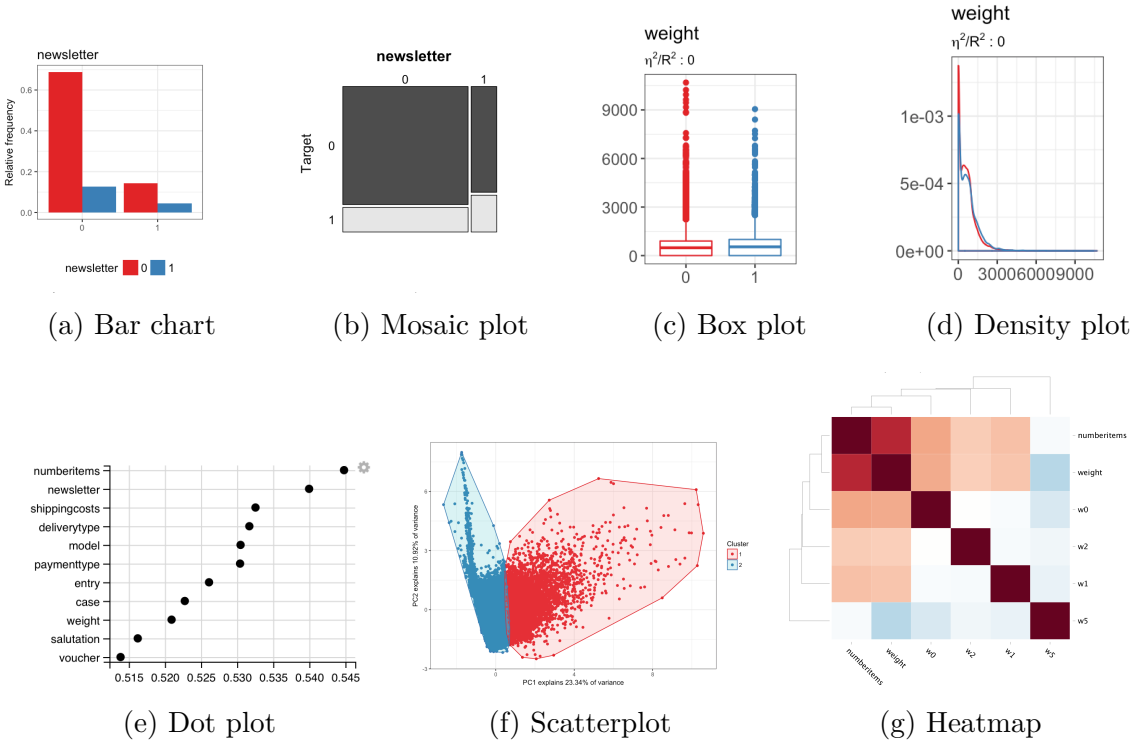
Box plots are suitable for numeric variables and appear in different variations. The basic idea of a box plot is to depict the central bulk of data with a box (e.g. from the first to the third quartiles), displaying a measure of location (e.g. median) inside the box, and stretching whiskers from the end of the box to a particular level (e.g. 1.5 IQR or minimum/maximum) (Unwin 2015, 44–47). **Kernel density estimators** can be viewed as smoothed histograms and tend to offer much better approximation of the density than widely used histograms (Pearson 2011, 355–365). Estimates are influenced by a kernel function K and a bandwidth parameter h. The former is usually a unimodal function with peak at 0 and it has less impact than value of h (Hand et al. 2001, 59–61). Parallel box plots (Figure 1c) and grouped density plots (Figure 1d) compare distributions of subgroups.

Cleveland dot plots (Figure 1e) are a visualization where labeled numeric values are depicted with dots positioned on a common scale. Under some conditions (e.g. baseline of zero), this plot is similar to a horizontal bar chart (Myatt and Johnson 2009, 43–44). **Scatterplots** explore the relationship between two continuous variables and may uncover associations, outliers, and clusters. A third variable can be introduced by color coding the plot points (Unwin 2015, 75–93) as in Figure 1f. The presence of massive amounts of data often decreases the usefulness of this graphical display because large numbers of points may conceal the association (Hand et al. 2001, 62–64).

Heatmaps are designed for visualizing a matrix with quantitative elements. Each value is mapped to a color spectrum. Cells are either colored relative to values in a column (e.g. in a matrix plot described in Section 3.2.2) or with regard to a fixed scale (e.g from -1 to 1 for a correlation matrix) (Piegorsch 2015, 102–105). Dendrograms (see details in Section 2.4.2) add clustering capabilities to rows, columns or both. This may reveal structures, but groups' interpretation is flexible and rather subjective (Piegorsch 2015, 102–105; Unwin 2015, 164–166). Figure 1g shows an example of a correlation plot with clustered rows and columns.

2.4 Unsupervised learning

The goal of unsupervised learning is to discover underlying structures of input data (often multivariate $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_p]$). In contrast to supervised learning, introduced in



(a) Bar chart (b) Mosaic plot (c) Box plot (d) Density plot

(e) Dot plot (f) Scatterplot (g) Heatmap

Figure 1: Examples of visualization from the developed software tool using Data Mining Cup (2010) dataset

Section 2.5, no dependent variable is present. Techniques such as principal component analysis (Section 2.4.1), self-organizing maps, multidimensional scaling, etc. seek to explore associations between multiple variables and whether the $\mathbf{X}$-space can be fairly well represented by a lower number of dimensions. Clustering techniques (Section 2.4.2) search for distinct groups of observations. The lack of clear quality measures of unsupervised analysis raises a challenge for their application. At the same time, these methods contribute to a better understanding of the data and are valuable for data exploration (Hastie et al. 2009, 486–487).

2.4.1 Principal component analysis (PCA)

The key idea of PCA is to express original p-dimensional data with a smaller q number of linear combinations of the p variables and to capture the highest possible variability (at maximum $q = p$). Principal components (PCs) can be derived either from the covariance or the correlation matrices. The latter is more practical due to invariability to scale of the original variables. Formally, $\mathbf{X}^*$ is a matrix of original variables scaled by their standard deviations. Then the covariance matrix of standardized $\mathbf{X}^*$ is the correlation matrix of the original variables. The goal is to find $\mathbf{a}_j = (a_{1j}, ..., a_{pj})$ such that for all $j = 1, ..., q$ the j-th PC is $\mathbf{u}_j = \mathbf{a}_j^\intercal \mathbf{X}^*$ with as high as possible variance subject to being uncorrelated with other $\mathbf{u}_j$.

It can be shown that $\mathbf{a}_j$s are the eigenvectors of the covariance matrix of standardized variables, ordered according to decreasing eigenvalues λ_j. An additional constraint is required to prevent inflation of variance due to larger component weights $\mathbf{a}_j$; a

usual choice is $\mathbf{a}_j$ having unit length ($\mathbf{a}_j^\intercal \mathbf{a}_j = 1$) with $Var(\mathbf{u}_j) = \lambda_j$. Consequently, coefficients lie between -1 and 1 and they reflect the contribution of original variables to a PC. Applying PCs to each observation forms PC scores (Clarke et al. 2009, 495–497; Piegorsch 2015, 342–345; Jolliffe 2002, 4–6).

There is no unanimity about the preferred criteria to choose a number of components. Possible strategies include retaining PCs corresponding to eingenvalues that are above the average. If standardize data is used, this is equal to keeping components that explain more variance than any of the original variables (or above 1). An alternative recommended threshold is 0.7. Another approach is to choose q such that a predetermined fraction of original variance is pertained, e.g. 80%. A scree plot of eigenvalues against the number of components demonstrates how quickly they drop to zero (Piegorsch 2015, 345–346). The first q PCs can be considered as dimensions embodying the "signal" and others containing the "noise" (Duda et al. 2001, 568–569).

A rotation of the axis within the q-dimensional space seeks to increase the interpretability of components. Formally, $\mathbf{A_q}$, a $p \times q$ matrix of loading, is multiplied by a $q \times q$ matrix $\mathbf{T}$. The choice of $\mathbf{T}$ determines the rotation type: orthogonal (e.g. varimax, quartimax) or oblique (e.g. promax, oblimin). The varimax rotation, for example, seeks to simplify columns of the loadings matrix, so that all variables have coefficients either closer to 0 or to 1. Details on rotations and the transformation matrix can be found in Jolliffe (2002, 152–154, 270–271), Larose and Larose (2015, 114–117) and Kotz et al. (2006, 11:7357–7357).

2.4.2 Clustering

Clustering techniques explore a dataset with an attempt to find natural groupings. The goal is to organize objects in such a way that instances within a group are similar to each other and different from objects in other groups. This requires quantitative measure of closeness or proximity (see details in Section 2.1.6).

The choice of dissimilarity measure is fundamental for clustering. It should reflect the analyst's understanding of closeness for a particular dataset. According to Hastie et al. (2009, 506), the specification of an appropriate proximity measure is by far more important to achieve good clustering results than the choice of an algorithm. Specification includes dissimilarity measures between attributes' values and their combinations (weighting) to calculate the proximity between individual observations. The distance measures for quantitative and mixed variables include weights w_j (see Equations (2) – (4)), which can be set explicitly or determined from the data.

Clustering techniques are often grouped into partition-based (optimization), hierarchical, and model-based clustering. The first type attempts to find optimal partition into K disjoint sets according to a chosen criterion (score) function. The second type seeks to discover a hierarchy of nested clusters and the partition is obtained by cutting at some level of the hierarchy. The third type is based on a probabilistic model for

clusters in a population and therefore not considered in this work.

Generally, when using unsupervised learning, the evaluation of the resulting clustering solution is not straightforward. Deciding on how many clusters are present in the data is problematic regardless of the chosen clustering method. An assessment in terms of the subject matter is crucial, but is not amenable for general research. Statistical evaluation is possible with external and internal techniques. External validity (e.g. Rand or Jaccard indexes) assesses similarity between two different partitions of the same data. Internal clustering criteria (e.g. Calinski-Harabasz, Davis-Bouldin, silhouette or gap indexes) attempt to evaluate how well clustering reflects internal structure (Clarke et al. 2009, 480–483; Everitt et al. 2011, 264–273).

Partition-based (optimization) clustering techniques attempt to assign observations to K disjoint groups. Once an inter-individual proximity measure is chosen, a criterion (or score) function that measures quality of a particular partition should be defined. The next step is to find a partition that maximizes or minimizes the criterion function.

Some score functions operate on the basis of the original data matrix, whereas others require inter-individual dissimilarities. While the former are limited to continuous variables only and often default to the Euclidean distance, the latter can accommodate all data types and proximity measures via a dissimilarity matrix. The general idea is to search for compact clusters that are far from each other. The clustering criterion can significantly influence the type and structure of clusters that are searched for (Everitt et al. 2011, 111; Hand et al. 2001, 299).

Criterion functions for numeric data are often used within cluster variation (W) to measure how tight clusters are and between cluster variation (B) to evaluate distances between clusters. A simple measure of W is a sum of squared Euclidean distances between a center of a cluster, expressed by its mean, and every data point in this cluster. B can be measured as distance between cluster means. Fortunately, minimization of W is connected to maximization of B, because they sum up to the total variation T. The sum-of-squares criterion (equivalent to a trace criterion, $tr(W)$) is criticized for its dependence on the scale of the variables. Other criteria based on B and W, including invariant ones, are available. Those, however, also have certain disadvantages (Duda et al. 2001, 542–548; Hand et al. 2001, 297–300). Clustering criteria for a dissimilarity matrix are based on the same notions of compactness and farness. A generalization of a score function for numeric data is to minimize arbitrarily defined dissimilarities to a cluster center expressed by one of the observations.

Once a score function is chosen, clustering becomes an optimization problem (maximization or minimization depending on the clustering criterion). The sample of observation is finite. Consequently, the number of possible observations' assignments to K clusters is also finite. Certainly, exhaustive enumeration is feasible only for tiny datasets; otherwise an iterative-improvement optimization is performed. The general

idea is to start with some clustering, reassign data points to improve the score function, find updated cluster centers, reassign points again, and repeat iterations until either no improvement of the criterion function, no change in the cluster membership, or a specified maximum number of iterations is achieved. Overall, only a fraction of possible points' allocations to K cluster is examined. Like any greedy search approach, such an algorithm does guarantee a local, but not a global maximum or minimum. Nevertheless, computational feasibility makes this approach attractive (Hand et al. 2001, 302–303; Duda et al. 2001, 548–550).

***K*-means** clustering is the most popular partition-based algorithm (a few versions exist) and can handle large quantities of data. It is appropriate if all variables are quantitative, squared Euclidean distance (in its standard version) is believed to reflect dissimilarities in the data and if it is acceptable that cluster centers are represented by a mean (a mathematical construct rather than real observation). The criterion function is the sum of squared deviations of points from the cluster means. As described above, as a rule, an optimization continues until convergence to a (local) minimum. The Hartigan–Wong algorithm additionally assures that no single reassignment of an observation to another groups will decrease the value of the score function. Under some conditions, this algorithm may not converge and an alternative, e.g. by MacQueen, can be used. Different starting points may influence the final solution (Hastie et al. 2009, 509–510). Repeating the procedure with different starting points provides some insight into the sensitivity to this parameter and allows one to increase the chances of attaining the global minimum. Clarke et al. (2009, 410) recommend using at least ten different sets of starting points.

***K*-medoids** clustering relaxes the limitations in terms of proximity measure choice. However, this comes at the expense of computation. This approach is applicable either to a data frame of attributes (numeric variables only) or a dissimilarity matrix (any type of variables). The minimization of a sum of squared distances in K-means is substituted by a minimization of the sum of defined dissimilarities to candidate representative objects (medoids).

Partitioning around medoids (PAM) is one of the K-medoids algorithms and goes as follows. First, K initial medoids are successively selected. The first object has the minimum sum of dissimilarities to all observations in a sample. All following representative objects (up to K) attain a maximum decrease in the criterion function. Given the initial set of K medoids all points are assigned to the nearest one, then one of the non-medoids is selected to be exchanged with one of the medoids and the score function is calculated. In case of improvement, points stay interchanged, while in the opposite case, the next non-medoid is swapped and so on until no single switch can decrease the criterion function (Kaufman and Rousseeuw 1990, 68–72, 102–104). This procedure does not scale up well, but it is less sensitive to outliers than K-means where

using squared Euclidean distance often causes extreme values to have high influence (Hastie et al. 2009, 515–517).

CLARA (abbreviated from Clustering LARge Applications) is a K-medoids partitioning method adapted to large amounts of data. In contrast to PAM, only a data matrix of numeric variables can be imputed. The algorithm version proposed by Kaufman and Rousseeuw (1990) calculates either Euclidean or Manhattan inter-object dissimilarities. A lowering of computational intensity is achieved via drawing a few (e.g. five) sub-samples of a fixed size, finding K clusters in each sub-sample, assigning each observation outside of a sub-sample to the nearest medoid and calculating the score function for the whole dataset. A solution with the lowest mean (equivalent to sum) of specified dissimilarities is chosen as a final solution (Kaufman and Rousseeuw 1990, 126–127, 144–146).

Internal validity criteria for different partitioning methods vary. Clustering based on the sum-of-squares score function deploys plot of W or $logW$ against number of clusters or a Calinski-Harabasz pseudo-F statistic: $F = \frac{(T-W)/(K-1)}{W/(n-K)}$. An "elbow" (or "kink") on such plot or a maximum value of the statistic may signal a good choice of K (Piegorsch 2015, 389–390). The gap statistic searches for the largest difference between $logW$ curve and a curve obtained from a suitable null reference distribution. Virtually, this is equivalent to searching for a "kick". In contrast to other methods, the gap statistic succeeds at detecting a single cluster solution if no groups are present in the data. Another advantage is its applicability to any clustering method (Tibshirani et al. 2001; Hastie et al. 2009, 519–520). Average silhoette width over the entire dataset is a scalar statistic derived from a silhoette plot. This graphical representation of clustering methods is based on notions of compactness and separation (Rousseeuw 1987; Kaufman and Rousseeuw 1990, 83–88). Although it was developed for partition-based techniques, it is applicable to hierarchical clusters, once the number of groups is decided.

In summary, the proximity measure should reflect the researcher's understanding of similarity, K can be chosen based on domain knowledge or experimentally (e.g. with the help of internal validation metrics), and the choice of an algorithm depends on the characteristics of the data and the goals of the analysis.

Hierarchical clustering algorithms build a hierarchy of clusters that implicitly contain any number of groups $1 \leqslant K \leqslant n$. This approach is different from the partition-based methods that determine a fixed number of clusters, even if no natural groupings exist. An analyst often tries various K to find an appropriate number of clusters. Notably, the same issue of constructing clusters even if no natural groupings exists is true for hierarchical clusters. Additionally, partition-based methods establish K clusters at once, while hierarchical methods build clusters using already existing ones. This requires an inter-group proximity measure in addition to inter-object dissimilarities that

were discussed above. Hierarchical techniques are subdivided into **agglomerative** (starts with n single-object clusters and then successively merges observations) and **divisive** (starts with n observations as a single cluster and then successively separate into smaller groups).

Given the inter-object dissimilarities, an agglomerative clustering algorithm merges two observations with the smallest inter-object dissimilarity and all successive merges are based on the defined measures of distance between clusters (see details for inter-group distances in Section 2.1.6, while recommendation of use can be found in Clarke et al. (2009, 415–417) or Duda et al. (2001, 552–555)). All fusions are irrevocable. The resulting structure is presentable with a dendrogram, a tree-like graph that traditionally displays the single-observation "leaves" ($K = n$) on the bottom and $K = 1$ on the top while y-axis quantifies inter-group distances. Drawing a horizontal line on a dendrogram "cuts" the branches to define a particular number of clusters. It is important to realize that this graph represents a clustering structure as obtained by a particular algorithm rather then the graphical display of the data itself (Hastie et al. 2009, 520–523).

2.5 Supervised learning: classification

The goal of supervised learning is to predict an outcome measurement ($\mathbf{y}$) from a set of p variables ($\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_p]$). A training set ($D_{train} = (x_i, y_i)$, $i = 1, ..., n_{train}$) is used to learn from a set of objects to then predict the outcome for unseen objects. $\mathbf{y}$ is variously referred to as the response, target or dependent variable and $\mathbf{X}$ as predictors, features, attributes, covariates, independent, explanatory or input variables. Learning is possible under an assumption that the measurement of interest is related to predictors in some way. If the target variable is categorical, the task of learning a mapping from $\mathbf{X}$ to $\mathbf{y}$ is called classification. Each y_i belongs to one of $C_1, ..., C_Q$ classes; in binary classification $Q = 2$. A zero-one loss function implies equal cost for misclassification in different categories and is used most frequently (Hastie et al. 2009, 9–21; Hand et al. 2001, 327–330). A learning algorithm is diversely referred as a model, classifier, or learner.

Many algorithms can naturally have more then two classes (e.g. decision tree based, k-nearest-neighbor), where as others need a special approach to adjust binary classification to Q categories (Flach 2012, 81–91, 102). While some classifiers output only class labels, other learners give a degree of certainty of belonging to a particular category (predicted probabilities). This work considers only the latter kind.

Classifiers are primarily evaluated by their generalization performance or prediction capability. Many researchers proclaim a trade-off between predictive power and interpretability (Hastie et al. 2009, 350–352; Clarke et al. 2009, 307–308; Shmueli 2010). The latter criterion is not a goal of the developed tool and hence it is not considered

here. In order to quantify generalization one should decide what to measure and how to measure it. Section 2.5.1 gives an overview of different performance metrics, Section 2.5.2 presents variations of dataset splits to estimate generalization ability, and lastly various classifiers are described in Section 2.5.3.

2.5.1 Performance measures

Performance measures are subdivided into threshold, rank, and probability metrics.[1] Both scalar and graphical measures are described.

Threshold measures are based on predicted categories of the outcome variable. Estimated probabilities of test instances are compared with a threshold c to predict class membership. In binary classification, an instance with predicted probability above the threshold is commonly assigned a prediction class of 1 while instances with predicted probabilities below the threshold are assigned a 0.

A cross-tabulation of the predicted and actual class membership (also called confusion matrix) is a usual way to summarize class predictions for any number of classes. In binary classification, a category of interest is denoted as a positive class. The diagonal elements in Table 1 correspond to correctly classified observations: true positive (TP) and true negative (TN). Misclassified events are called false negatives (FN) and incorrectly predicted nonevents – false positives (FP). Table 2 contains a list of metrics based on a confusion matrix.

Observed	Predicted	
	Event	Nonevent
Event	TP	FN
Nonevent	FP	TN

Table 1: Schematic confusion matrix for $Q = 2$

Metric	Description
TP rate (TPR, sensitivity, recall)	$TP/(TP+FN)$
TN rate (TNR, specificity)	$TN/(FP+TN)$
Positive predictive value (PPV, precision)	$TP/(TP+FP)$
Negative predictive value (NPV)	$TN/(TN+FN)$
Accuracy	$(TP+TN)/(TP+TN+FP+FN)$
Balanced accuracy	$0.\dot{5}(TPR+TNR)$
Cohen's κ	$(O-E)/(1-E)$ where O is the observed accuracy and E is the expected accuracy based on the marginal totals

Table 2: Performance metrics based on a confusion matrix for $Q = 2$

The choice of an evaluation measure depends on the problem at hand and reasons for including specific measures are discussed in Section 3.2.7. In short, the first four

[1] This taxonomy appear in Japkowicz and Shah (2011, 315) and in Ferri et al. (2009).

metrics from Table 2 have a single-class focus and are mostly used in binary classification, while the remaining metrics concentrate on all classes. Balanced accuracy in the provided form is defined for binary classification only. Cohen's κ (details and examples can be found in Everitt 1992, 146–150) and accuracy calculated from a $Q \times Q$ confusion matrix assess the performance of a multiclass learner.

Rank measures assess the ability of a classifier to rank observations by their probability of class membership. These metrics are independent of calibration and scaling. A graphical tool for assessing performance over all possible thresholds – a receiver operative characteristic (ROC) curve – shows the estimated proportion of correctly classified events against the estimated proportion of nonevents predicted as events. The resulting curve inside a unit square shows a perfect classifier if it goes vertically from a (0, 0) to (0, 1) point (all positives correctly classified, no mistakes in negatives), while a diagonal line refers to a chance classifier (Hand et al. 2001, 361; Piegorsch 2015, 296–297; Japkowicz and Shah 2011, 113–117). The area under curve (AUC) is an attractive scalar measure of performance, which is independent of a chosen threshold. However, ROC curves of compared classifiers often cross and a model with higher AUC can be inferior in a specific region (Hand 2009).

While ROC analysis for binary classification is an established analytical method, extension of AUC to multiclass case is not straightforward and resulted in many variants. Japkowicz and Shah (2011, 131) recommends using a generalization by Hand and Till (2001) based on the AUCs of all possible pairwise binary classifiers.

Probability measures reflect a classifier's confidence about predicted classes. In contrast to threshold metrics, probability measures account for classification uncertainty, e.g. observation with estimated probabilities of 0.51 and 0.99 would belong to the same class in case of $c = 0.5$, but deviations from the true probabilities evaluate these two observations differently. Brier score quantifies these deviations as mean squared error (Japkowicz and Shah 2011, 106–107; Kuhn and Johnson 2013, 247–248; Ferri et al. 2009).

2.5.2 Resampling methods

Once an analyst has chosen performance metrics that best suit a dataset and problem at hand, they should decide on what data these performance measurements are calculated on. A naive approach of using a complete dataset for training and for evaluation results in an overoptimistic estimate of a model's generalization ability. Importantly, two distinct goals of model selection (choice of tuning parameters of a classifier) and model assessment (how a final model does when applied to the unseen data) can be of interest. In a best case data-rich scenario, one can split a dataset into equal training, validation, and test fractions of data, where the train set is used to train classifiers with different tuning parameters, a validation set is used to select a best learner among them and finally, a test set assesses the performance of a chosen model (Hastie et al.

2009, 222–223). If no tuning is performed then measuring performance on the test set suffices.

A **holdout** set is probably the simplest measuring protocol. This method splits the entire sample into two mutually exclusive sets, where one is used for training and another for model assessment. The fraction of data left for testing depends on the overall number of observations; a quarter, a third or a half of a sample are typical amounts (James et al. 2013, 176–177).

The ***K*-fold cross-validation** (CV) method divides available data into K (roughly) equal subsets (folds) and applies a classifier K times, each time treating one fraction as a test set and using the remaining $K-1$ sets for training. Consequently, all data is eventually used for statistical learning and averaging K performance estimates reduces their variability (Hastie et al. 2009, 241–242).

Balancing datasets with considerably low relative frequency of a response class (rare event) allows a classifier to learn about all kinds of records, not primarily the high frequency category. In case of binary classification, the simplest balancing methods include oversampling/upsampling and undersampling/downsampling. In the former, instances of minority class are randomly sampled to achieve a more balanced distribution (e.g. increase fraction from 7% to 30% or 50%), while in the latter, the number of majority class records is reduced. Importantly, the test set is not balanced in order to imitate the new data (Larose and Larose 2015, 166–167).

2.5.3 Predictive models

The **logistic regression** models probabilities of Q classes via a linear function of the explanatory variables. In cases of binary classification, only one linear function is estimated. In short, estimation is performed via iteratively (re)weighted least squares, maximum likelihood algorithms, or feed-forward neural networks. Details on the first two more traditional approaches can be found in Hilbe (2009, 51–61) or Hastie et al. (2009, 120–121) while the last method is described in Ripley (1996, 145–153).

The ***k*-nearest-neighbor (*k*-NN)** is a distance-based learning method. Each test observation from $D_{test} = (x_i, y_i),\ i = 1, ..., n_{test}$ is compared with covariates of train instances, then k "nearest" are selected and a test instance is labeled by a majority class among the neighbors (Hastie et al. 2009, 463–465). A weighted k-NN technique offers a modification of the basic nearest neighbor algorithm: it transforms calculated distances into weights according to a kernel. This function of the distance, d, has a maximum at $d = 0$ and decreases as dissimilarity grows. While a rectangular kernel assigns equal weights to all k instances, other kernels assure that more distant training cases have less influence on the prediction. The predicted probability for a particular class of a new instance is proportional to the sum of the weights for this category among the k train instances. In summary, the weighted k-NN requires a specification of the measure of similarity (see details in Section 2.1.6), a number of neighbors k, and a kernel function.

As for other distance-based procedures, data for k-NN is standardized to ensure equal weights of all predictors. Numeric attributes are divided by their standard deviations and categorical variables are binarized, and consequently standardized, according to a procedure described in Hechenbichler and Schliep (2004).

The **decision tree** approach recursively divides the feature space into disjoint rectangles. At first, the "root" (complete dataset) is split into two groups by the most informative feature that can achieve the most homogeneous samples. Splitting criteria can be chosen by classification error rate, Gini impurity, or other alternatives. Full recursive partitioning would result in absolutely pure nodes (with instances of the same class), sometimes of single observation. This is likely to overfit the train data and give poor prediction on the test set. Stopping rules may include minimum size of node or a threshold for decrease in impurity. Alternatively, an algorithm grows full trees and then prune it back ("cut branches"). Depending on the observed features' values, each new instance lands in one of the terminal nodes and a proportion of the majority class in this node is assign as a predicted probability (Piegorsch 2015, 312–321; Hastie et al. 2009, 305–310). Further details and examples of classification trees can be found in Clarke et al. (2009, 249–253), Flach (2012, 129–147) and Duda et al. (2001, 395-413).

The **random forest** is a generalization of recursive partitioning that combines multiple decision trees. This approach averages over single learners which typically decreases variability and often results in a better classifier (Clarke et al. 2009, 254–255). Two main components of random forests are bagging (bootstrap aggregation) and random selection of features. The former involves building each tree from a sample of size n_{train} drawn with replacement. The latter consist of limiting the number of attributes available at each split to $m \leq p$. In combination, this enables the exploration of the predictive power of all features and creates "de-correlated" trees. The probability prediction for a new instance corresponds to a proportion of trees that result in a particular class (Hastie et al. 2009, 587–593).

3 Software

This section presents a data analysis tool developed for the Data Science team of SHS VIVEON AG and named Data Discovery Platform (DDP). The terms "tool", "app", and "software" are used interchangeably. A discussion of the app's goals (Section 3.1) is followed by a detailed description of its functionality and technical implementation (Section 3.2) as well as the app's deliverables (Section 3.3) and user interface (Section 3.4). The end users contributed to the content and appearance of the software, and some of their comments are mentioned throughout of the app description. User research methods and more general results of their application are presented in Section 3.5.

3.1 Goals

The business application for the software is the exploration of tabular data received from clients. A classification task implies a categorical target variable and one or more predictors. A typical dataset contains a few dozens of thousands of observations and a few dozens of variables. The target group for the tool is members of the Data Science team all of whom are specialists with considerable knowledge of data analysis.

Aside from tackling the general business goal of decreasing the costs involved in completing repetitive tasks, the tool was specifically created to:

1. provide a first impression of the information in the data and conduct a data sanity check;
2. assess the appropriateness of client's data for a prediction classification task (binary or multiclass); and
3. allow users to analyze data interactively, quickly, and without any programming effort and function intuitively for a person with a solid statistical and machine learning background.

The first two goals suggest a combination of data exploration and initial predictive analytics. The data needs to lend itself to classification and the software tool seeks to assesses how adequate the data is for given task. Working with secondary data from a client raises a potential quality concerns, e.g. attributes that have no relation to the target variable were collected or the same level as a categorical predictor coded in different ways (e.g. "male" and "M"). In addition, reports of the analysis should be available to support decision making, e.g. whether the consultancy firm can work further with the provided data.

The third goal addresses the interests of both the company and end users. While the employer values the speed and quality of the performed tasks, the users benefit from the interactivity, graphical interface, and automation, allowing them to focus on interpretation rather than the correctness of typed commands.

3.2 Description and technical implementation

Data characterization appears as descriptive numeric summaries, graphical characterizations, and data models (Pearson 2011, 6). The tool's functionality represents techniques of all these categories and is enriched by the detection of groups in the data via clustering. Table 3 summarizes techniques included in DDP.

The software tool is constructed as a series of panels (tabs) dedicated to particular topics, e.g. clustering. All tabs are presented in Sections 3.2.1 – 3.2.7 and a general view of each panel can be found in Appendix A. Although a navigation bar (on the left on Figure 2) offers soft guidance about a sequence of steps, the analysts are encouraged to follow their intuition, particularly because discoveries with one technique may spark interest to return to an already completed analysis, e.g. substitution of extreme

Function	Tools
Data preparation and preprocessing	Data import, change of variable type, detection of constant and almost constant predictors, identification of highly correlated predictors, exploration of extreme and missing values
Description, visualization	Plots of distributions, measures of association, PCA, variable importance rating
Grouping	K-means, PAM, CLARA, and agglomerative hierarchical clustering
Prediction	(Multinomial) logistic regression, k-NN, decision trees, and random forest with different resampling methods

Table 3: Summary of techniques implemented in the software

values by missing values suggests that analysts should return to examining patterns of "missingness". Besides navigation, the sidebar contains fields with a target variable and *Excluded variable(s)* widget for (temporary) elimination of attributes from the analysis.

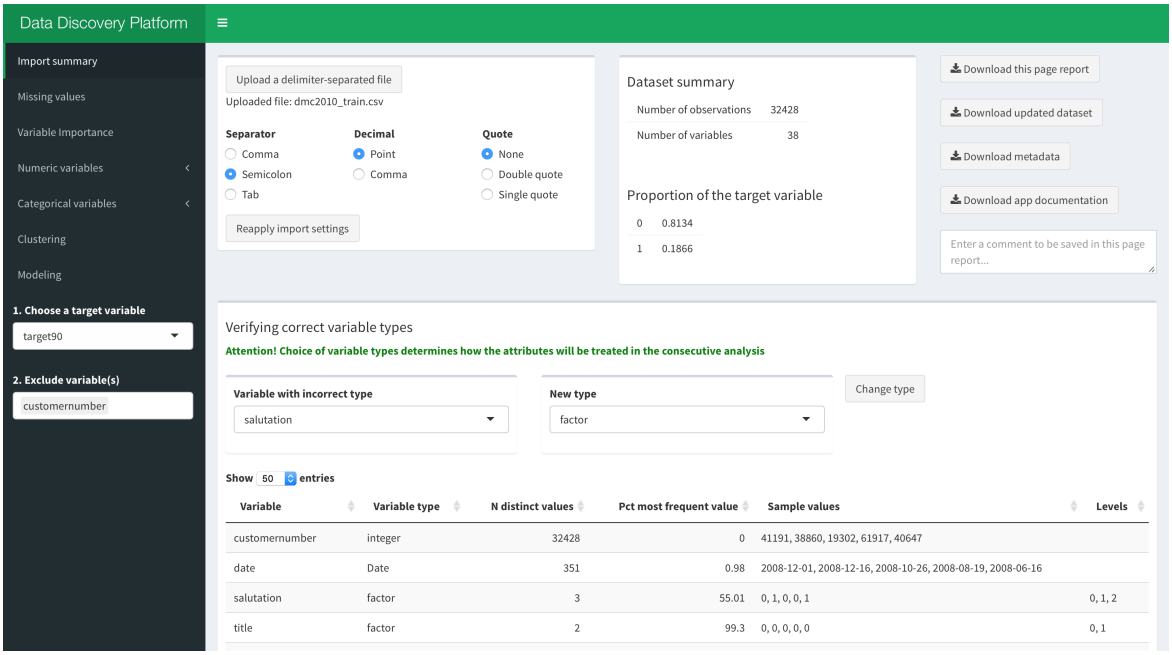


Figure 2: General view of the software tool

Each panel of the software contains a button to download that page report and a comment field that is transferred into the report (located in the top right corner of Figure 2). The comment option allows analysts to save their insights within the software. When appropriate, panels contain in-app notes about the number of observations and variables that were used in a particular analysis. This information is to assist users whose confidence about results may depend on the size of the data on which a technique was applied

The software functionality relies heavily on visualizations that have a potential to elucidate quantitative information and communicate it simply yet powerfully (Tufte 2001, 9). Multiple plots per page encourage and facilitate comparison, and expose similarities and distinctions. Due to the possibly large number of charts of individual attributes, the app prioritizes predictors with estimated stronger classification ability. This facilitates perception and additionally saves end users' time. Statistical graphics depicting separate observations are useful if the number of cases is not too large. This restriction motivates the design choice of giving preference to graphics summarizing distributions, such as box plot and kernel density estimators (as described in Section 2.3). Numerical statistics often accompany statistical graphics in order to support the possible visual discovery.

In order to achieve the goal of having interactive interfaced-based controls, the appropriate technical based solutions had to be employed. A so-called shiny app enables access to R (R Core Team 2016) capabilities for statistical analysis through an interactive web interface. The advantages of using R include a wide and ever expanding range of cutting-edge statistical and machine learning methods, a vital supportive community of users, and its free-of-charge basis (Crawley 2013, xxiii). Intensive use of this scripting language by members of the SHS VIVEON's Data Science team simplifies future modifications of the tool. Moreover, ready-to-use building blocks from **shiny** (Chang et al. 2016) and **shinydashboard** (Chang 2016) packages permit construction of a web interface without any CSS or HTML scripting skills. Technical limitations such as single thread processing and the impossibility of stopping a running R routine without app termination did not outweigh the benefits.

At a later stage of the project the software tool was turned into a **ddp** package and installed at a SHS VIVEON's server. The version control is enabled via the company's BitBucket. This solution was motivated by proprietary nature of the created code and therefore prohibition of publicly available tools as GitHub. The core of the **ddp** package contains *global.R* (load of packages, auxiliary functions and some definitions), *ui.R* (structure and components of user interface), *server.R* (server side of the application where calculations are defined), multiple *<name_of_report>.Rmd* for downloadable reports, and *DDPfunctions.R* (manually created auxiliary function for supporting repeated operations and making code more readable).

3.2.1 Import summary

The *Import summary* tab is a landing page of the app that includes upload capabilities and presents summaries of the dataset. Figure 19 portrays the general view of this tab. Assumptions about an input file include delimiter-separated format (e.g. .csv, .txt), presence of variable names in the first row and settings among the available ones (comma, semicolon, tab as column separator, point or comma as decimal sign and none, double or single quotes). The upper limit of the file (200MB) approximately cor-

responds to the maximum dataset size processable by the app at an acceptable speed. In case of a large dataset, a sample can be explored in the software. The file upload is enabled by `shinyFileChoose()` from the **shinyFiles** package (Pedersen 2016). Then the file is read in `read_delim()` from the **readr** package (Wickham et al. 2016). This function was selected for its ability to automatically parse dates which is not available in a more common `read.csv()` from the **utils** package (R Core Team 2016).

The dataset is summarized in a table that lists all variables with their corresponding type, number of distinct values, percentage of the most frequent value, minimum and maximum values (for nonnumeric fields maximum and minimum string), as well as a few sample values, and levels if a variable is cast as categorical. The table is implemented with `datatable()` from the **DT** package (Xie 2016) and can be sorted by any column to ease the review of information about the variables. The inclusion of the number of unique values and the percentage of occurrence of the most frequent value is motivated by the implications of discreteness of numeric variables (e.g. in Spearman rank coefficient or resistant outlier detection). The inclusion of these elements can also support decisions about variable type in case of missing or incomplete metadata. Minimum and maximum values may point out anomalies, e.g. negative height or human age of in years of 999.

Supported variables types include `factor`, `integer`, `numeric`, `character`, and `Date`. Once a dataset is fed into the app, an auxiliary function `AssignClasses()` converts all variables with values either 0 or 1 into nominal variables. Full automation of type detection is not feasible, e.g. integers can encode categories or have numeric meaning. The analyst is encouraged to verify the correctness of the variable types and modify them if necessary as it determines how the attributes are treated in the subsequent analysis. Changes can be done via the interface: *Variable with incorrect type* drop-down contains all variables, *New type* drop-down gives all possible variable types, and *Change type* button completes the change, which is immediately reflected in the summary table.

The user has multiple ways to verify that data was successfully uploaded. First, drop-down widgets contain lists of variables. Second, a table with a number of cases and variables reflects data dimensions. Third, the summary table displays a list of variables and their sample values. The early version of the app required a second upload in the case of wrong upload settings. This was mentioned as undesirable during the usability tests and currently the user can adjust and reapply settings (e.g. change decimal mark) with a *Reapply import setting* button.

The distribution of the target variable is displayed as soon as it is chosen in a *Target variable* drop-down. The inclusion of this information into the first page emphasizes the classification focus of the software. In the case of highly unbalanced classes, an analyst can adjust their judgments about results in the later stages of the analysis.

3.2.2 Missing values

The *Missing values* tab provides a numeric summary of missing values per variable (univariate inspection) and offers a visualization designed to help the analyst to understand patterns of "missingness" (multivarite exploration). The table orders variables by percent of missing values so that the analyst can concentrate on predictors where the issue is most widespread. Additionally, the user can limit the number of displayed rows via *Show variable with % of missing values over* slider widget. The general view of this tap is depicted in Figure 20.

Missing values in a dataset are depicted with a matrix plot, a heatmap (see details in Section 2.3) where each row represents an observation and each column – a variable, resulting in each value of a dataset to be depicted by a rectangle (Templ et al. 2012, 41–42). Missing values are marked red, while filled cells are colored by a continuous color scheme for which the values are scaled to an interval [0,1] (Templ et al. 2016, 37). Although some patterns become visible, data that is systematically missing variables may exhibit a sporadic-looking missing data pattern (Enders 2010, 4–5). By default, `matrixplot()` from the **VIM** package (Templ et al. 2016) graphs observations as rows and variables as columns, except the excluded ones, in their original order. Observations can be sorted according to a variable with a *Sort matrix plot by* drop-down widget. The visualization in Figure 6 is sorted by `deliverydatereal` and demonstrates some irregularities in other variables for rows with missing values.

The matrix plot was chosen as a final solution for various reasons. Firstly, typical dataset sizes used by the Data Science team are depictable in this way. Specifically, the app renders a plot for datasets with fewer than 300,000 observations which was experimentally defined. Otherwise, an in-app note about the maximum number of observations is displayed. Secondly, a heatmap of a dataset is a more intuitive graphical display than its aggregated version depicting frequencies (see detailed description in Templ et al. 2012, 34–35). Thirdly, aggregation is connected to loss of information. In fact, sorting by a variable in the matrix plot is analogue to aggregation and reveals "co-missingness" of values in different variables.

Imputation, mentioned in the theoretical background in Section 2.1.3, is not implemented in the app due to the software goal of the first impression and sanity check of raw data. The only exclusion is an implemented option to create an additional `missing` level in factor variables with the help of `fct_explicit_na()` from the **forcats** package (Wickham 2017).

Missing values in predictors are correctly detected as such, if entries in the uploaded data matrix appear as blank fields, `NA` or `NaN`. The data dictionary or any other documentation may reveal the specifics about a dataset, e.g. missing values in variable age are marked with 999. In such cases, these values should be replaced with blank fields or `NA` prior to analysis. Additionally, a lack of proper metadata can result in a

problem of disguised missing values when numeric codes are treated as observed values (e.g. 0 for BMI in the Pima Indians diabetes dataset, see Pearson 2011, 88–92). Such values are not automatically detected by DDP as missing, but a researcher may notice anomalies in other steps of EDA and correct it.

3.2.3 Variable importance

The *Variable importance* tab (Figure 21) visualizes ranking of attributes by their ability to predict the target variable, measured by ROC AUC (see theoretical background in Section 2.5.1). This approach includes building classifiers with a single independent variable, (multinomial) logistic regressions with 3-fold cross-validation, and measuring predictive performance on test data. Such univariate screening was preferred to separate rankings for numeric and categorical variables since all attributes are comparable.

The technical implementation is similar to training this type of classifier and calculation of either binary of multiclass AUC in the *Modeling* tab. This process is encapsulated in `CalculateUnivariateAuc(data, target, balance)` auxiliary function that outputs a data frame ready for plotting. The `balance` argument takes one of three possible values for use of original, over- or undersampled data.

A Cleveland dot plot (as described in Section 2.3) is chosen for mainly two reasons. Firstly, it is compact, has high data-ink ratio and its labels are easily readable, which is especially important if the number of variables is large. Secondly, a dot chart may outperform other graphs (e.g. horizontal bar chart) thanks to the human ability to compare objects' position on a common scale (Cleveland 1984, 270–275; Myatt and Johnson 2009, 43–44). The plot is implemented with `ggvis()` from the **ggvis** (Chang and Wickham 2016), so that hover over on the points displays a value of ROC AUC.

3.2.4 Numeric variables

The *Numeric variables* tab is subdivided into preprocessing, correlation plots, PCA, and plots. Preprocessing allows the user to clean extreme values, detect constant and nearly constant predictors, and explore pairwise correlations. Real-valued variables (`numeric`) and integer-values count variables (`integer`) are analyzed in this tab.

The transformation of individual explanatory variables is a frequent preprocessing step and indeed it has the potential to improve predictive performance of classification algorithms. However, the application of these techniques should be carefully tied to the data. Because of this, the transformation was considered excessively complex for automation and unnecessary for an initial exploration of the data. A second type of transformations, mentioned in Section 2.1.5, is carried out according to specific methods, which are described in their respective sections below.

Preprocessing: Extreme values

An analyst can choose either to deal with extreme values in the data preparation stage or to apply procedures that will be less sensitive to them. The former includes

the option of detecting extreme values and then either omitting them, replacing them with plausible values or examining them separately (Pearson 2011, 287–294). However, the absence of reliable and universal detection methods makes this path challenging (Sprent 1998, 64). Likewise, practical difficulties arise while using resistant estimators. The use of measures capable of mitigating the influence of extreme values is not always effective and, consequently, the detection of such values may be essential (Behrens 1997, 144; Yu 2010, 11). For these reasons, the app offers several methods for the detection and omission of extreme values as well as some analytical routines that are relatively insensitive to them.

Both versions of the Equation (1), one with mean and standard deviation and another with median and MADM, are implemented in the software. This is because reasonable agreement between traditional and resistant methods of outlier detection strengthens confidence in the outcome, while greatly differing results are likely to indicate an unusual data structure (Pearson 2011, 276). Consequently, the end users benefit from the comparison of the two procedures and can choose the one that they see as more sensible for the analyzed data. Furthermore, as each of the approaches has its own advantages and disadvantages, one of them may be more appropriate for a particular dataset. In absence of the normality assumption, a version of a decision rule with mean and standard deviation can reflect a belief of a nontypical value that in distant from the center of the data. This method, however, is flawed by so called masking: both mean and standard deviation are sensitive to extreme values and these metrics mask the same their presence. Alternatively, median is a measure of location and MADM estimator is a measure of scale, both of which are insensitive to extreme values. As in the case with the mean/standard deviation rule, values of a variable can be declared extreme if their magnitude exceeds a specific number of MADMs of that variable. The MADM's breakdown point has the highest possible value of about 0.5 and is advantageous for outlier detection (Wilcox 2010, 32–33). While this procedure is much less sensitive to masking, the opposite problem of declaring some legitimate points outliers (swamping) may occur (Pearson 2011, 305). On the negative side, MADM is zero if the fraction of the most frequent value exceeds 50%. In such case, all values unequal to the median would be considered extreme for any t.

The value of t should be selected by a user via the *Number of standard deviations* or *Number of median absolute deviations* slider widget respectively. Different values in a range from two to six are suggested by researchers (see an overview in Pearson 2011, Section 7.5.3) and three is chosen as the widget default. General view of the *Extreme values* tab is depicted in Figure 22.

Due to a possibly high number of variables, no detailed information per predictor is offered. Instead, a note about the remaining number of complete cases is displayed. This supports the user's decision regarding whether to apply the substitution of ex-

treme values with `NA` and, if so, what value to choose for t. On a technical note, detection and substitution of extreme values by missings is implemented with auxiliary functions `FindOutliers(data, t)` and `FindOutliers.resist(data, t)`.

In summary, a quick solution for inspecting extreme values sets a number of sample spread measures (standard deviations and MADMs) for all variables. The app also allows to inspect numeric variables one by one or in small subgroups if other variables are temporarily excluded via the *Excluded variable(s)* widget. Detection and substitution of extreme values by some plausible values as well as detection and further scrutiny were considered too complex for an initial examination of a dataset.

Preprocessing: Constant and almost constant predictors

Constant fields have no explanatory power to a response variable. They may emerge out of inspection of a fraction of data (e.g. panel data for one year only), signal problems in data collection (e.g. failure of a sensor), etc. Regardless of the source, such variables are noninformative in a dataset and, moreover, hinder some analytical procedures (e.g. result in division by zero in the Pearson correlation coefficient).

Almost constant (near-zero variance) attributes are also candidates for exclusion. The first reason for this is that analysis is in essence interested in explaining changes in a target variable in response to variation in an explanatory variable (Pearson 2011, 97–98). The second reason is that resampling at the modeling stage (see details in Section 2.5.2) may result in chunks of data with constant columns.

Constant and almost constant predictors are detected with `nearZeroVar()` from the **caret** package (Kuhn 2016). Two tuning parameters from this function are implemented in slider widgets *Frequency ratio* (`freqCut`) and *Percent of unique values* (`uniqueCut`). The details of these parameters are discussed in Section 2.1.2. The output table presents only variables with zero and near-zero variance in order to avoid superfluous output (see an example in Figure 23).

Preprocessing: Highly correlated predictors

Similar to the *Extreme values* panel, this part of the software offers users the option of choosing less and more resistant measures to encourage comparison. The less resistant measures are represented by Pearson product moment correlation coefficient while the more resistant measures are represented by Spearman's rank correlation coefficient. An output table presents pairs of correlation and shows only those exceeding a threshold determined in the *Absolute value of correlation* widget for at least one of the metrics. Details on the measures of association included in the app can be found in Section 2.2 and the general view of this tab is presented in Figure 24.

Pearson correlation coefficient is probably the most extensively used association measure for numeric variables. The end users explicitly asked for its display in the tool. However, it is very sensitive to extreme values, e.g. association can be missed because of one anomalistic observation (sample breakdown point is $1/n$) (Wilcox 2010,

123, 169–172; Pearson 2011, 450–459). For this reason, it was decided that this well known metric would be presented alongside more resilient methods.

Spearman's ρ and Kendall's τ_b are based on values' ranks and reflect the strength of a monotone relationship rather than just a linear one. While Kendall's τ_b explicitly accounts for concordant pairs, Spearman's coefficient can suffer from ties and may not be advisable for highly discrete numeric variables. The implementation of Kendall's τ_b was prevented by its low calculation speed for large datasets.

Correlation coefficients are calculated by specifying the method `"pearson"` and `"spearman"` in the `cor(..., use = "pairwise.complete.obs")` function from the **stats** package (R Core Team 2016). The results are undefined if at least one of the variables is constant. This condition is checked prior to calculation and, if needed, a help message is displayed. The user can detect zero-variance predictors in the *Constant and almost constant variables* in *Preprocessing* and exclude them from analysis via the *Excluded variable(s)* widget.

In theory, global measures of association would offer an alternative way to decrease sensitivity to extreme values. Metrics such as the minimum volume ellipsoid or the minimum covariance determinant estimators take the overall structure of a dataset into account (Wilcox 2010, 180–186). However, these measures are less researched and end users of the software tool are not likely to be familiar with them, which would make interpretation challenging. Therefore, the global measures of association are not included in the app.

Correlation plots

Patterns are often more obvious from a plot than raw values, especially for larger matrices. Hence, it is advantageous to present matrices of pairwise correlations visually. At the same time, a heatmap relies on distinguishing differences in color hue and saturation, the perception of which is believed to be relatively inaccurate (Cleveland and McGill 1984).

The user chooses whether to plot Pearson correlation coefficients or Spearman's ρ via a radio button widget (with Pearson as the default). The general view of the tab is shown in Figure 25. For the above mentioned reasons, constant predictors should be excluded. Correlations are visualized by being clustered according to correlation strength and their sign, and presented in two plots, one for almost constant predictors (as detected in tab *Constant and almost constant variables*) and another for all the others. This split into two plots lowers the number of variables in each visualization, which may contribute to easier perception. Moreover, this allows the users to support their decision about whether to exclude almost constant predictors from further analysis.

The app uses a modified version of `d3heatmap()` from the **d3heatmap** package (Cheng and Galili 2016) with a hierarchical clustering of rows and columns of the

correlation matrix. This technical solution is able to accommodate large correlation matrices thanks to hover over information about a pair of variables and a correlation coefficient. The function modification designates the range of colors from -1 to 1, from dark blue to dark red with white corresponding to 0, and to use a colorblind safe palette of 11 colors from the **RColorBrewer** package (Neuwirth 2014) (extrapolated to 220 by `colorRampPalette()` from the **grDevices** package by R Core Team 2016).

PCA

While other *Numeric variables* tabs consider one variable at a time or their pairs, principal components analysis examines all variables at once. PCA explores associations between multiple attributes and can signal the amount of structure, detectable by linear combinations of the original variables. It reduces complex data into a simpler representation with the hope of elucidating patterns. The PCA implemented in the app is based on a correlation matrix. Further background on the PCA is given in Section 2.4.1.

The *PCA* tab (Figure 26) contains a scree plot, a table with eigenvalues and cumulative proportion of variance explained and a heatmap of coefficients (loadings). The PC scores are added as columns to the downloadable extended dataset (variable names include counting number of a component with a prefix, "PC" if no rotation was done, "RC" in case of orthogonal rotation `varimax` or `quartimax` and "TC" in case of oblique rotation `oblimin`).

A scree plot elected to be used instead of the other methods for choosing PCs (see Section 2.4.1) because it allows users to make a visual assessment of the data rather than relying on a numeric threshold. The plot provides an understanding of which components capture most of the structure. The flexibility of assessment is advantageous as it does not tie an analyst to a specific cut-off (e.g. explain 80% of variance), which cannot be known in advance and is not a focus of the analysis.

Carrying out a PCA for exploratory purposes raises the question of the results' interpretability. Unlike factor analysis, this method is not deployed to discover latent factors. Nevertheless, rotations can simplify the interpretation of loadings, but not without drawbacks. The value of cummulated explained variance remains, but it is redistributed from the highest PC to the others and can lead to missing a dominant component. Moreover, results after rotation are very sensitive to the choice of q (see more details and examples in Jolliffe 2002, 269–272).

While PCA can increase understanding of a dataset, it has its limitations. This technique considers only linear combinations of the original variables and is based on the first two moments. This technique is of limited use for data with nonlinear structures or those in which distributions are better characterized with higher moments (Clarke et al. 2009, 494). Moreover, the resulting projection is not optimized in terms of predictive performance and sometimes clouds the class differences in datasets intended

for classification (Hand et al. 2001, 196).

On the technical side, a scree plot is created by `fa.parallel(..., fa = "pc")` and PCA is performed by `principal()` from the **psych** package (Revelle 2016). This function takes as inputs the number of components and rotation, which are specified by the user in widgets *Number of components* and *Rotation* respectively. Possible rotations include `none`, `varimax`, `quartimax`, and `oblimin`.

In general, the functionality of the software allows users to run PCA, download an extended dataset, load it again and feed the PC scores into a classifier. This may be sensible if strong correlation structures are present and/or dimensionality of the original data is large. This can be the next iteration of the analysis and initially, the correlation structure is just assessed.

Plots

This tab (Figure 27) presents parallel box plots and grouped kernel density plots of individual numeric attributes, conditioned on the level of the target variable. Different graphical displays are included because of their ability to highlight different attributes' characteristics. Multiple box plots contain less information about distribution than kernel density estimates. However, the box plots' simpler form often conveys information with greater clarity than kernel density estimates, especially when the target variables have many categories (Hand et al. 2001, 59–61). The use of the target variable as a grouping factor and ordering of plots according to ROC AUC calculated in the *Variable importance* tab promote exploration of attributes without losing sight of the overall classification goal.

The visualizations are created with the **ggplot2** package (Wickham 2009) and each level of target variable is distinguished by a color from a colorblind safe "Set 1" from the **RColorBrewer** (Neuwirth 2014). The data for plotting is prepared with `melt()` from the **reshape2** package (Wickham 2007) and display of multiple plots is facilitated by `grid.arrange()` from the **gridExtra** package (Auguie 2016).

A box plot summarizes the distribution of a variable in a few statistics, and it assesses asymmetry, dispersion, and extreme values. As described in Section 2.3, the borders of the box correspond to 25th and 75th quantiles, the line inside the box corresponds to median, and the whiskers stretch to 1.5 IQR (Wickham 2009). Box plots display specific values of empirical distributions, but the variation between them, e.g multimodality, is obscured by this type of graph (Unwin 2015, 44–47). Other visualizations of numeric variables' distribution, such as histograms and kernel density estimators, can reveal these variations.

Kernel density estimators of a variable at different levels of the factor (grouped density plot) show the differences in distributions between classes (area under each curve is 1, independent of the subgroups' size) as well as the distribution's shape in each subgroup (e.g. modality, skewness). The density estimates are positioned in

one plot to encourage comparison. The kernel parameter K, mentioned in Section 2.3, is specified as Gaussian. Experimentation with different bandwidths showed wide applicability of the default of the `density()` function. In contrast, estimation of the Sheather–Jones rule (`bw = "SJ"`), recommended by Venables and Ripley (2002, 128–129), failed for some data.

Visualizations are accompanied by an effect size for the strength of association between a numeric attribute and the target variable, η^2 (see details in Section 2.2). In the special case of one numeric and one categorical variable, the value of this measure of association is equal to the coefficient of determination R^2 in a simple linear regression with one categorical variable. The fact that end users may not be familiar with η^2 necessitates that its value be displayed with a comment of equality to R^2. The association measure is calculated with `summary(lm(attribute~target))$r.squared` from the **stats** package.

3.2.5 Categorical variables

Due to the focus on classification, categorical predictors (type `factor`) are considered in relation to the target variable. The general view of this tab is depicted in Figure 28. The distributions are visualized by clustered (grouped) bar plots and mosaic plots (as described in Section 2.3). Graphics are ordered by ROC AUC, as calculated in the *Variable importance* tab.

A clustered (grouped) bar chart benefits from having the highly familiar bar format. It compares one variable at different levels of another (Larose and Larose 2015, 59). Differences in the length of the bars are easily perceived by the eye, which allows this graphic to convey data adequately (Cleveland and McGill 1984, 533). Grouped bar plots are created with `ggplot()` from the **ggplot2** package. The y-axis depicts the relative frequency with the total number of observations corresponding to 100%.

The usability tests of the software tool showed that users are unacquainted with mosaic plots. Nevertheless, this alternative visualization of contingency tables is included in the tool, primarily for its ability to communicate dependency. In particular, a similar fraction of one variables at different levels of another forms an "unbroken" straight line between rectangles and indicates the absence of dependency. On the contrary, a "broken" line points out dependency (Unwin 2015, 131–137). Mosaic plots are created with `mosaicplot()` from the **graphics** package (R Core Team 2016).

The aggregation of less frequent levels in a nominal attribute into an "other" category is performed if the number of categories exceeds four. This prevents overcrowding in the plots and allows the analyst to concentrate on the most populous categories. At the same time this may conceal repetitive categories with different names (e.g. "M" and "male").

Visual displays are supported by measures of association between two categorical (nominal) variables with two or more levels. As a dataset can contain variables with

different numbers of categories and their comparison is desirable, a more universal measure of association, Cramér's V, is chosen for the reasons outlined in Section 2.2. An asymmetric version of Goodman and Kruskal's λ with the target variable as a dependent one is well suited for the app's focus on classification. The resulting metric reveals the percentage by which knowledge about a categorical attribute reduces the probability of error (Everitt 1992, 56–59). Software implementation calculates the sample measure by `assocstats(table(attribute, target))$cramer` from the **vcd** package (Meyer et al. 2016). Goodman and Kruskal's λ is calculated with a auxiliary function `calc.lambda.col(table(attribute, target))` adjusted to output only λ for predictions of target from another variable (Schwartz 2015).

3.2.6 Clustering

The *Clustering* panel contains subtabs for four clustering methods included in the tool: K-means, CLARA, PAM, and agglomerative hierarchical clustering (Figure 29). There is also a tabular overview of these methods. The theoretical background and details of these techniques can be found in Section 2.4.2. The application of a clustering method to a dataset consists of two steps:

- *Step 1 (optional): Choosing an optimal number of clusters* presents the results of internal validity criteria (Figure 30); and
- *Step 2: Performing cluster analysis* presents the results of clustering: group sizes and various visualizations of variables included in the clustering process (Figure 31). Simultaneously, a variable of cluster membership is added as a column to the downloadable extended dataset.

The four algorithms were chosen to address different input data requirements (size, type of variables) and to enable a comparison of results. K-means is probably the most well-known clustering method. Its strength lies in its ability to handle large amounts of data relatively quickly, an important feature for many datasets. The following characteristics can be a downside to K-means, depending on the project goals and the data: K-means only handles numeric data and its basic version is restricted to squared Euclidean distance (as defined in Equation (2), squared), K-means requires that decision about the standardization of variables be made and extreme values can negatively influence the outcome, the K-mean's optimization algorithm cannot guarantee the global extremum, K-means does not create a hierarchical structure of groups, and no missing values are allowed.

PAM is less sensitive to extreme values than K-means for the same distance measures. It can handle any data types via a dissimilarity matrix and accepts missing values with rare exceptions, which are described at the end of Section 2.1.6. The clusters are represented by real observations in contrast to mean values in K-means, which is not a clear advantage or disadvantage. PAM's inability to handle large data is a potential drawback, which CLARA is solving. However, this partition-based method

for large datasets uses subsets to find clusters and this does not necessarily fully reflect the overall structure. It is also limited to numeric variables. In order to cover different views of the data, Manhattan distance (as defined in Equation (3)) is predefined in CLARA clustering.

Divisive hierarchical methods tend to require more computations and there is no clear evidence of their superiority in comparison to agglomerative methods (Piegorsch 2015, 384). For these reasons, agglomerative clustering tends to be more commonly used in practice and has been implemented in the software. The software includes three types of inter-group distances: average, complete, and single linkages (see details in Section 2.1.6). These distances determine the set of clusters from which a final solution is chosen. Other measures, such as centroid linkage or Ward's method, that build on the notion of means, are not applicable to arbitrary dissimilarities (Everitt et al. 2011, 76–78) and, therefore, are not implemented in the software tool.

Clustering methods are executed on preprocessed data. K-means and CLARA algorithms input numeric variables that are normalized prior to clustering in order to prevent attributes with large numeric values from dominating distance calculations. This normalization is equivalent to using variability weights. Although standardization is widely applied, it may conceal the groupings that were well separated before the transformation (Everitt et al. 2011, 43–69; Duda et al. 2001, 538–541). A min-max transformation is implemented in the app, because it showed relatively good performance in experimental studies. At the same time, they postulate that appropriate standardization method depends on the data and other clustering parameters. A simulation study examining different standardization procedures on datasets with varying percentage of contaminated data and separation level of clusters by Milligan and Cooper (1988) showed that standardization involving scaling by the range of a variable is favorable to other types of standardization. A study concentrating on K-means clustering also showed that max-min transformation performed better than z-scores (Steinley 2004). Moreover, Kaufman and Rousseeuw (1990, 6–11) also warn against scaling by a standard deviation that is highly influenced by extreme values. Technically, normalization is implemented with `data.Normalization(..., type = "n4", normalization = "column")` from the **clusterSim** package (Walesiak and Dudek 2016). PAM and hierarchical clustering are based on a dissimilarity matrix from both numeric and categorical variables by Gower's generalized coefficient, as described by Equation (4), implemented with `daisy(..., metric = "gower")` from the **cluster** package (Maechler et al. 2016). However, computational challenges prohibit the calculation of such matrices for large datasets.

Step 1 of each clustering method provides means for choosing K. The meaningful partition of data can be checked by a domain expert, but the end user of the app is not likely to be an expert in the field pertinent to the client data. This increases the

importance of offering internal validity criteria in the software. In step 1, the K-means, PAM, and CLARA algorithms are run repeatedly with different numbers of clusters, from one to a number specified by the user in the *Maximum number of clusters* widget. The results are plotted and displayed as in-app notes. The agglomerative hierarchical clustering is constructed once for each linkage and then the "branches" are cut to form different numbers of groups, as in other clustering methods, from one to the user specified maximum. In addition to plotted internal validity criteria, dendrograms for each linkage are displayed.

Internal validity criteria vary with different clustering methods. The silhouette internal validity criterion is implemented for all four algorithms and visualized with a plot of average silhouette width against different numbers of clusters. This measure is assessed via `$silinfo$avg.width` in `pam()` and `clara()` results or calculated with `silhouette()` from the **cluster** package for K-means and hierarchical clustering. The calculation uses their corresponding dissimilarity matrices as input. Researchers give some guidance to this statistic interpretation: values under 0.25 signal no substantial structure, whereas values over 0.7 indicate a strong structure with other levels in between (Kaufman and Rousseeuw 1990, 83–88). Additionally, Davies-Bouldin's index (`index.DB()$DB` from the **clusterSim** package) and Calinski-Harabasz pseudo F-statistic (`index.G1()` from the **clusterSim** package) are calculated for each K in K-means and the number of clusters corresponding to the smallest Davies-Bouldin's index and to the largest Calinski-Harabasz pseudo F-statistic is outputted.

The gap statistic was included in the software to assess whether any groups exist during the initial analysis of the data. This may be crucial, because as mentioned in Section 2.4.2, the clustering methods build clusters even if no natural groupings exist. The gap statistic is implemented with the `clusGap(..., B = 100)` from the **cluster** package. Due to the fact that this function accepts only numeric attributes, the app demonstrates this statistic only for K-means and CLARA clustering. The reference null distribution is left at its default value (`spaceH0 = "scaledPCA"`), as described in the original paper by Tibshirani et al. (2001). Due to computational considerations, the number of bootstraps generating the reference distribution is lowered to 100 instead of the recommended 500. Moreover, if the number of rows in the dataset exceeds 5,000, a random sample of this size is drawn for the statistic calculation. The `clusGap()` includes parameters for specific clustering methods (e.g. number of sets of starting values in K-means). The parameters resemble the specification of the applied clustering techniques in Step 2. The optimal number of clusters is chosen by the rule proposed in the original paper (`maxSE(..., method = "Tibs2001SEmax")`). During the development of the software, this internal validity criterion was tested with a simulated dataset with "noise" attributes and indeed, the gap statistic recommended one cluster.

Step 2 requires the specification of the number of clusters and, in addition, a linkage in case of hierarchical clustering. There is intentionally no default in this radio button widget, because suitability of an inter-group proximity measure depends on the data. Step 2 presents cluster sizes and visualizes the input variables. Importantly, the attributes are shown on their original scale, irrespective of the standardization method used for clustering. Numeric variables are visualized by parallel box plots, kernel density estimators, and for K-means, in addition, by bar plots of means. The first two types of visualizations are created like the visualization in *Numeric variables* tab described in Section 3.2.4 with the only difference being that cluster membership is a grouping factor. Bar plots of means are created by `ggplot()` and depict means of standardized variables with corresponding error bars of one standard deviation. Categorical variables are represented as grouped bar plots by cluster membership. The ordering of attributes by variable importance allows users to compare with distributions of variables on different target levels in *Numeric variables* > *Plots* tab.

A *Clusters in relation to principal components* plot is a scatter plot (as described in Section 2.3) of the first two principal components with color-coded cluster membership of observations. This representation depicts information in the numeric variables at a glance, as opposed to depicting the individual variables, which is especially interesting if the number of fields is high. However, this low-dimensional representation is not guaranteed to reflect the presence of clusters (Everitt et al. 2011, 273–275). In particular, the kind of structures that can be revealed by this plot may be limited due to orthogonality of PCs (Jolliffe 2002, 78–80). While the PCA was calculated locally for this plot in the early versions of the app, the PCs are currently generated in the *PCA* subtab of the *Numeric variables* tab, where the user may adjust the number of PCs and rotation. These changes are immediately reflected in the plot displayed in the *Clustering* tab.

On the technical side, clustering is performed whenever computationally feasible with the following procedures. The function `kmeans(..., iter.max = 50, nstart = 25)` from the **stats** package attempts to use the Hartigan-Wong algorithm with 25 different starting points; if it fails, MacQueen is used instead. Multiple sets of starting points are used to increase the chances of attaining a global minimum for a score function. CLARA is implemented with `clara(..., metric = "manhattan")` and PAM with `pam()` from the **cluster** package. Hierarchical clustering is performed with `hclust()` from the **fastcluster** package (Müllner 2013). Sourcing function `hclust()` from this optimized package improved performance in comparison to a same-name function in **stats** package. An experiment of constructing dendrograms with three different linkages on a dataset of 10,000 observations demonstrated that average speed (in 100 runs) dropped from 10.3 to 5.4 seconds.

Some methods may be more appropriate for particular data and certain types of

analytic objectives. It may be reasonable to apply different methods, compare their solutions, and assess their appropriateness. Similarity in clustering results gives some confidence in the presence of natural groupings in a sample. In contrast, drastically different solutions may signal the absence of clear sub-classes (Everitt et al. 2011, 257–260). The possible use of different numbers of cases due to differences in handling missing values discourages the application of an external validity criterion (as described in Section 2.4.2). If needed, this analysis can be done with cluster membership information from a downloaded extended dataset.

Clustering is included in the software, primarily, because the identification and interpretation of groups increases understanding of the data. Clustering can also be used as a preparation step for building a classifier. In particular, strong differences encourage experimenting with separate predictive models for each group. In this case, however, the test instances should first be clustered (Myatt and Johnson 2014, 84). This scenario can be considered by an analyst as a result of clustering. However, separate models are not directly supported by the software. Besides, if clustering results in groups that are highly similar to the target categories, the definition of the response variable may be reevaluated. Often, the binary classification, e.g. into either churn or non-churn (0/1) depends on a definition, e.g. churn withing x months.

3.2.7 Modeling

The *Modeling* tab provides users with their a first glimpse at the predictive power of the data. It offers a choice of the dataset (original, under- or upsampled), the resampling method (holdout or cross-validation), the positive class (for binary classification only), and the different classifiers. The output is concentrated on presenting predictive performance on test data, numerically and graphically. Presenting the result of any classifier in a consistent format facilitates comparison, and the performance of any two models can be compared in a separate tab. Due to this emphasis on prediction, the output intentionally does not include coefficients and similar model outcomes. The results are displayed in subtabs for *Model 1*, *Model 2*, and *Model comparison* (see Figure 3). Additionally, an example of the output of a learner is presented in Figures 33 and 34. The theoretical background on supervised learning, assessing performance, resampling methods and specific models can be found in Section 2.5.

Learners of different classes, degree of simplicity, and scalability are included in the software. The current version includes four classifiers: (multinomial) logistic regression, k-NN, decision tree, and random forest. Logistic regression is a technique well-knows to end users. It makes rather strong assumptions about the data and includes at least p parameters. Following the terminology of machine learning, it is a stable classifier. Despite its simplicity, k-NN is believed to be successful if boundaries of the classes are irregular (Hastie et al. 2009, 463–468). Decision trees based methods implicitly perform variable selection and binning of numeric variables, which mitigates

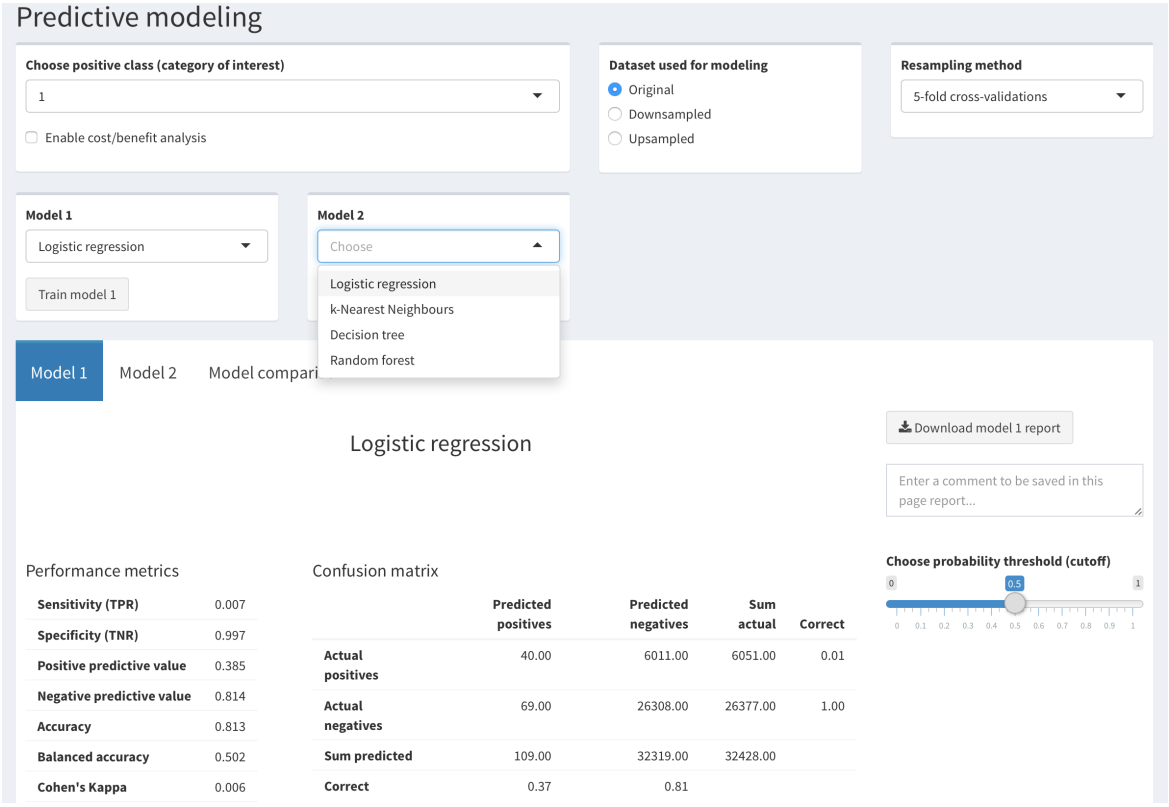


Figure 3: General view of the *Modeling* tab

the potentially harmful effects of extreme values. A single tree has the further advantage of low computation cost. At the same time it is characterized as an unstable high variance learner, i.e. "slight" changes to the data can "largely" change predictive performance (Hastie et al. 2009, 350–352; Duda et al. 2001, 475–476). Random forest was included because of its reputation for having high predictive performance on many datasets and also due to its relatively low number of tuning parameters in comparison to other ensemble methods (e.g. gradient boosted model). No model is always the best and the performance results of different models can give an idea about the type of relationship between the target and predictors, e.g. if k-NN shows a good result in comparison to logistic regression, the relationships are likely nonlinear.

Experimenting with some simple learning methods without tuning parameters can bring to light some of the complexities inherent in the data. The software's capabilities are deliberately limited based on the assumption that parameter tuning and data transformation (e.g. binning) have the potential to improve performance.

The **mlr** package (Bischl et al. 2016) covers all functions implemented in this section. In short, `makeClassifTask()` specifies data, target variable and positive class (in binary case). The function `makeResampleInstance()` creates an index of observations that belong to train and test sets using the input from the *Resampling method* drop-down widget. The command `makeLearner()` takes the name of a classifier as input and indicates to predict probabilities of an observation to belong to each class

(default probability threshold is $1/Q$ where Q is a number of classes). The outputs of the three functions (task, resampling instances and learner) are taken into `resample()` and predictions on the test dataset are made. The classifiers are implemented with `classif.multinom` (logistic regression), `classif.kknn` (k-NN), `classif.rpart` (decision tree), and `classif.randomForest` (random forest). Details and learner-specific default parameters are mentioned in Ripley and Venables (2016), Schliep and Hechenbichler (2016), Therneau et al. (2015), Breiman et al. (2015) respectively.

Early versions of the app trained different models with the same sampling procedure, but not necessarily with the same data splits. An intermediate step of the resampling instances stores information about train and test observations. This ensures fair model comparison that is not influenced by randomness due to the resampling for different models.

The software includes three kinds of performance metrics: threshold, rank, and probability (see Section 2.5.1 for details). The users have the option of using the metric they believe to be best suited for the data and project goals, or to use multiple metrics and compare the results.

Threshold measures include sensitivity, specificity, positive and negative predictive values, balanced accuracy, Cohen's κ, and accuracy (the last two are also calculated for multiclass classification). In order to calculate these measures, predicted probabilities are transformed into discrete categories. Class predictions are useful in many practical applications, e.g. spam or ham emails. In general, threshold measures are appropriate when an analyst is assessing the classifier's ability to separate classes with minimal error (Ferri et al. 2009, 27).

For binary classification, interactive cutting of predicted probabilities into classes determines point estimates for the values in the confusion matrix. Technically, the function `setThreshold()` is governed by a slider widget *Choose probability threshold (cutoff)*, as depicted in right part of Figure 3. Graphical displays of performance measures against the probability threshold (generated by `generateThreshVsPerfData()`) are exploratory in nature and complement point estimates of a confusion matrix. The analyst's decision about the cutoff is supported by line plots of performance metrics against a probability threshold. Additionally, depicting metrics in pairs (true positive and negative rates, positive and negative predictive values) shows the trade offs between them. Three accuracy measures in one graphical display demonstrate discrepancies between metrics with and without chance correction (see examples in Figure 4).

Sensitivity and specificity are accuracy measures for events and nonevents in the observed positives and observed negatives, respectively. This makes them conditional metrics or per-class accuracies, e.g. 90% sensitivity is the classifier's accuracy, assuming the email is non-spam. Unconditional statements are possible and expressed with

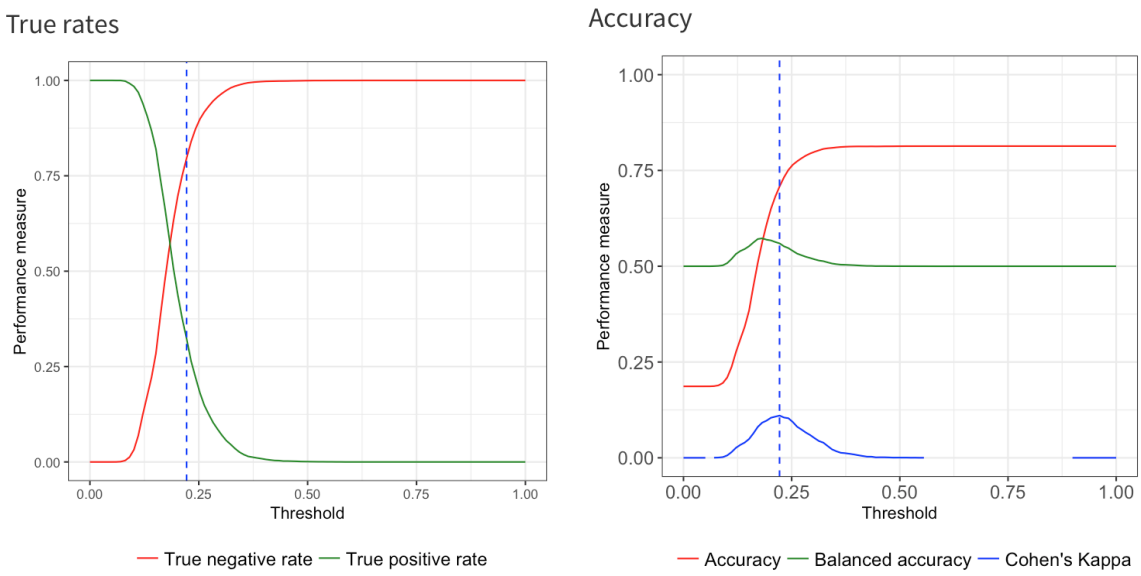


Figure 4: Performance metric vs. threshold plots with vertical dashed line corresponding to value from the *Choose probability threshold (cutoff)* widget

positive and negative predicted values. However, their estimation is difficult due to dependence on the prevalence of the event (Kuhn and Johnson 2013, 256–259). In fact, many researchers assume a fixed prevalence of 50% and then PPV and NPV can be calculated from values of a confusion matrix (Japkowicz and Shah 2011, 99–100).

Accuracy is intuitive to interpret, but is misleading in class-imbalance cases. One way to overcome overoptimism about a classifier is to compare the resulted accuracy with a no information rate (always predicting the majority class). Balanced accuracy would coincide with the conventional counterpart if a classifier performs equally well in both classes. Cohen's κ statistic, introduced as a measure of agreement between raters (Cohen 1960), compares observed accuracy with accuracy achieved by chance. Values between 0.3 and 0.5 are believed to signal reasonable agreement between actual and predicted classes (Kuhn and Johnson 2013, 254–256; Japkowicz and Shah 2011, 86–92).

Consequences of the correct and incorrect predictions can be quantified with a custom cost/benefit metric. This would normally require domain expert knowledge and understanding of business objectives. End users of the app would supposedly have access to domain experts on the client side. Assignment of costs and net benefits to all cells in a confusion matrix allows calculation of total or average profit (Kuhn and Johnson 2013, 260–262; Larose and Larose 2015, 460–463). Demonstration of costs and net benefits for different correct and erroneous predictions facilitates understanding of modeling results for statistical non-experts and can foster initial exploratory results for SHS VIVEON's clients.

While the **mlr** package offers a wide range of performance measures, two metrics were developed:

- For binary classification, `costvalue.measure` is created with `makeCostMeasure()` and inputs from the cost matrix (visible if the box *Enable cost/benefit analysis* is checked, see an example in Figure 32). The user assigns different costs to wrongly predicted observations and net benefits to correctly predicted ones. The resulting matrix is then multiplied with the four cells of the confusion matrix to calculate the total value for predictions on the test dataset.
- `kappaAC` is created with `makeMeasure()` taking in a function that calculates Cohen's κ of the confusion matrix with `cohen.kappa(table)$kappa` from the **psych** package (Revelle 2016).

Rank measures are presented by a ROC graph with the chosen threshold marked by a dot and the AUC scalar measure. The Brier score represents a numeric probability measure. However, richer information about model performance may be obtained from a visualization of predicted probabilities distribution for different true outcomes in the form of kernel density plots. An "ideal" classifier produces non-overlapping density estimators for true events and nonevents. A vertical line indicating a probability cutoff facilitates the choice of the threshold (Kuhn and Johnson 2013, 251–252; Hosmer et al. 2013, 174–176). For binary classification, the graphical display is implemented with `ggplot()` density of predicted probabilities at two levels of the true outcome. Importantly, this graph shows model's strengths and weaknesses for each class.

All the above mentioned metrics are calculated on test data. The app does not give an option to get performance on the training data, which is probably overoptimistic. Instead it offers various ways of splitting: holdout with test dataset of 1/2 or 1/3 or cross-validation with five or ten iterations. Stratification achieves roughly equivalent class distribution in train and test sets and is recommended, especially in the case of unequal proportions of the target variable (Flach 2012, 350).

The holdout set approach is simple, intuitive, and relatively fast (a classifier is trained only once). In general, learning with more data is preferred, but computational considerations can make initial training with a smaller fraction of data more attractive. However, the holdout set approach has at least two drawbacks. Firstly, a fraction of data excluded from training may contain important patterns. Secondly, resulting performance metrics vary considerably, depending on the cases that are included in the train set (James et al. 2013, 176–178).

K-fold cross-validation attempts to combat both of the disadvantages of the holdout set approach. The extreme case of $K = n$ is called leave-one-out CV and is characterized by high variance due to almost identical training sets. This possible drawback and high computational burden motivated the exclusion of leave-one-out CV from the software tool. Despite lacking a theoretical justification, 5- or 10-fold stratified CV are believed to give good results (Kohavi 1995; Hastie et al. 2009, 242–249; Japkowicz and Shah 2011, 202–204) and are implemented in the app.

The user is offered a choice among the original data, under- or oversampling, because balancing of datasets frequently shows improved classifiers' predictive performance (Härdle et al. 2009). Experimental studies regarding different balancing methods give contradictory results that could have been heavily influenced by a particular classifier and dataset (Batista et al. 2004). Finding no unanimity among researchers motivated the implementation of the two simplest methods – upsampling and downsampling for binary classification. Both techniques achieve an approximately equal percentage of cases for both classes in the train set, while preserving the original distribution in the test set. Importantly, upsampling increases the total number of observations and should be applied with caution for originally large data with a strong inbalance. On the other hand, downsampling randomly removes cases of the majority class, which may result in leaving out or obscuring the patterns. Subject to data size constraints, an analyst is encouraged to experiment with different balancing methods. In the software, the original dataset is a default that can be changed via the *Dataset used for modeling* radio button widget. If balancing is chosen, an `makeUndersampleWrapper()` or `makeOversampleWrapper()` is applied to the created with `makeLearner()` classifier.

3.3 Outputs / Deliverables

Deliverables of the tool are a documentation of the achieved results and provide a basis for further analysis. File names include a dataset name and a time-stamp to ensure versioning. An example of a set of outputs can be found on the enclosed CD (see a list of files in Appendix B).

The downloadable files incorporate:

- an updated dataset that includes columns of PCA scores and cluster membership, and reflects the substitution of extreme values by `NA`, if these steps were executed (.csv file);
- metadata that includes information on the variables level: variable type, importance score, percent of missing values, etc. and whether a variable was excluded from analysis (.csv file);
- result reports of individual pages that contain the final state of the performed actions in each tab and can be enriched by end user comments (.html files, generated by **markdown** (Allaire et al. 2016) and **pander** (Daróczi and Tsegelskyi 2015) packages); and
- general documentation that provides references to the implemented R routines and settings (.html file).

An advantage of using the .html format for the reports lies in maintaining some functionality that could not be presented in a .doc or .pdf. For example, this includes hover-over tooltips with information on correlation plots. In contrast to the interface where display is sometimes conditioned by the chosen view, reports contain all results

from a tab, e.g. both Pearson and Spearman correlation plots. The general documentation is included in the software in order to make it self-containing. Additionally, an internal company wiki-article was created mainly for promotional purposes. It outlines functionality and provides instructions on where and how to access the software tool.

3.4 User interface

Engagement with a digital product is possible and enabled through an interface. Many researchers emphasize that good design starts with an understanding of human behavior, not with layouts and sketches. The interface designer must first clarify the target audience's goals and intuitions to understand the behaviors relevant to the development of future software. The search for a visual solution should only start after outlining the main tasks and objects (Tidwell 2011, 1–9, 25). The target audience of the DDP app are data scientists of SHS VIVEON AG, which is a rather homogeneous group in terms of their technical and methodological training. The goals, outlined in Section 3.1, determine the organization of elements and choice of controls.

In terms if its form, the DDP app is a web application, so use of web conventions (e.g. evidently clickable elements or download icon) is beneficial. Familiar and recognizable interface parts make interaction easier and more enjoyable (Tidwell 2011, xvii; Krug 2014, 14). In terms of content and functionality, the app enables interactive exploration of a new dataset.

The app's global navigation is positioned on the left, which is conventional for websites. Signposts such as tab highlighting and page titles help the user to understand their location (Tidwell 2011, 77–85). The above mentioned design principles also provided the motivation to structure the navigation bar with a sequential list resembling an EDA workflow.

Each page is dedicated to one topic (each of them presented in Sections 3.2.1 – 3.2.7 with general views depicted in Figures 19 – 32). The grid divides a page into clear regions and it guides the user on what parts can be ignored and what to focus on (Saffer 2007, 122–123). The left panel of the DDP is visible all the time, which emphasizes its significance. Besides navigation, the left panel contains the target variable and excluded attributes widgets. When the user opens a new tab, titles indicate what one can expect from this page, and what tables or graphs are to be rendered. The largest central part of a page is reserved for outputs and all controls belonging to a particular page are positioned on the right. Only the *Import summary* and *Modeling* pages contain some interactive elements that are integrated with the output elements. In the former case a drop-down menu allows the user to change the variable type, which then should be confirmed by pressing a button (see Figure 19). The result is reflected in the table located underneath. In the latter case, this page sets parameters for two models in different subtabs (see Figure 32). In other words, tuning elements

positioned in the main panel has an effect on a number of steps, never solely on the current page. Consistently placing interactive elements either in the top area or on the right "teaches" a user where to look for tunable parameters and buttons to perform analysis steps.

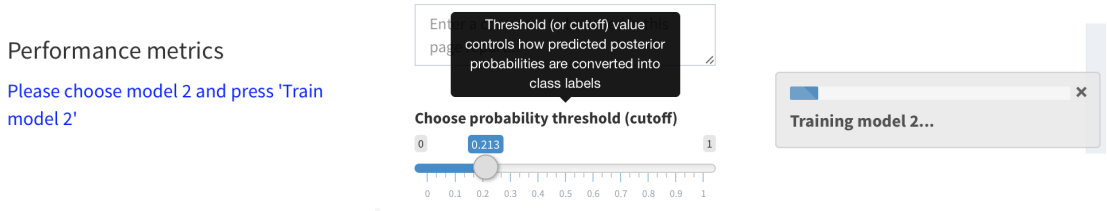
Assisting information appears as help messages in the output field, tooltips (hover over) for widgets and results, and progress bars. Their design and substance was influenced by user research, which showed that short, timely, and noticeable hints are preferred to instructions (Krug 2014, 47–51; Tidwell 2011, 364–368).

Help messages contain the smallest amount of information capable of guiding the user to the next step. They appear in a sequence, prompting users about each of the steps that are required before a particular output can be seen (see an example in Figure 5a). For example, the graphs are usually sorted by ROC AUC, which means that this calculation should happen before plots are displayed. In turn, a target variable should be selected for assigning AUC values. Besides improving user experience, these messages display errors or silent errors that **shiny** produces when objects cannot be rendered. The color blue was found to be adequate for help messages as it is distinct from the general green/gray palette, but has no connotation with errors, which are displayed in red. On the technical side, help messages in the output fields are implemented with `validate(need(logical condition, "message"))` from the **shiny** package. If a condition is not fulfilled (e.g. no target variable chosen), then a corresponding message appears. Validations are often piped to display them in a sequence.

Tooltips allow users to reduce visual noise and to make efficient use of screen space. Descriptive guiding information about the widgets and outputs appears when and where the user needs them while he or she is interacting with the app (see an example in Figure 5b). The initial solution for tooltips used a **shiny** function `helpText()`. The usability tests demonstrated that they appear too slowly to be consistently discovered by users. Use of functions from the **shinyBS** package (Bailey 2015) overcame the lag problem. Moreover, the appearance of these comments is customizable and they are more visually appealing. This change in technical implementation improves the users' experience when they have a doubt and require assistance. The `addTooltip()` is used for short notes about widgets (e.g. explains near zero variance cut offs), while `addPopover()` annotates the output (e.g. gives description of balanced accuracy).

Finally, the status/progress bar (see Figure 5c) indicates when and what procedure is running. The Data Science team members are active R users, which implies that they are used to be informed about a calculation being in process. Moreover, a procedure can be interrupted with a stop sign. As mentioned earlier, interfacing with R through a **shiny** app has the disadvantage of making it impossible to stop a procedure. Nevertheless, the user is informed about the currently running process. Time-consuming

operations are wrapped into `withProgress()` from the **shiny** package. This function displays a predefined message, e.g. "Calculating K-means cluster indicies...", and a progress bar. In case there is a loop procedure, information about the k's iterations is displayed with `incProgress()`, e.g. "Number of clusters: 4".



(a) Help message (b) Tooltip (c) Status bar

Figure 5: In-app assisting information

Users' actions are restricted by widgets and controls: buttons (binary action, e.g. download updated dataset), radio buttons (e.g. linkage method), drop-down menus (e.g. models), multiple selection lists (e.g. *Excluded variable(s)*), sliders (e.g. value of ROC AUC), spin boxes (numeric field and small buttons to increase or decrease value, e.g. number of principal components), etc. The choice of values, plausible ranges and defaults were chose to balance ease of use and flexibility.

No action can be directly undone, but most of the steps can be performed repeatedly with different settings. The only exception is the substitution of extreme values by missings and turning missing values in nominal variables into an additional category, which are irreversible. The inability to cancel an operation mentioned in section 3.2 is reflected in the user interface. Unfortunately, a user finds no button or keyboard shortcut to interrupt a time-consuming operation. Possible side effects of long waiting is likely to hinder active experimentation and exploration (Tidwell 2011, 269–274).

The interface was designed not only to serve current users, but also to allow future amendments. The horizontal tab structure in the *Preprocessing* and *Clustering* is easily extendable if the implementation of new methods is required.

3.5 User research

The research methods adopted in the development of the DDP are qualitative in nature and focus on understanding user behavior. These research methods are:

- focus groups, which is a way to discover opinions, wishes and needs of the target audience based on their experience with the existing product or reaction to a new action; and
- usability tests, which are a way to detect problems that confuse and frustrate users, and consist in the observation of how a user performs typical tasks and interacts with a web site, electronic devices, etc. (Krug 2014, 113, 119; Kuniavsky 2003, 201–207, 259).

Scholars find that qualitative research methods are more suitable for revealing user behavior, context and limitations of a product than quantitative techniques. Moreover, qualitative methods tend to be less expensive, faster, require smaller sample sizes, and can be adapted midway through the process. Direct observation, focus groups, individual interviews, usability testing and a combination of these activities are useful at different stages of a product's design (Goodwin 2009, 54–57; Cooper et al. 2007, 49–57).

The user research was an integral part of the software development project. The plan started with the creation of a schedule and budget. The first presentation of the DDP app and a focus group session took place a month after the start of the development. Usability testing and further group discussions occurred four and five months into the development.

The focus groups provided more general feedback and functioned as brainstorming sessions. Initially, the DDP handled only binary classification, but one of the group discussions concluded a need for covering multiclass problems as well. Group dynamics can be difficult to manage: some people participate in group discussion with reservation, talk drifts to unproductive topics, etc. (Goodwin 2009, 56). Despite the usefulness of a group discussion for general ideas, specific design questions are better understood in usability tests.

Usability tests were adopted due to their flexible nature and rich results even with a low number of testers (Krug 2010, 19–22). Actual future users performed usability tests in two stages with four sessions each. Each session lasted for about an hour. Users were encouraged to analyze their own datasets and to think aloud. Live observation and assessment of screen and voice recordings resulted in a list of comments, grouped into themes about each tab, performance issues, sidebar, miscellaneous and those that do not require any changes to the software. Next, comments were prioritized and worked through, if required. Issues were discussed in short follow-up talks with the Data Science team members.

Some users appreciated repeated visualizations and were comparing distributions of attributes in regard to the target variable and to cluster membership. The presence of alternative graphical displays, such as density estimators and box plots, was acknowledged as desirable, because they show data differently and because some analysts prefer one of the visualizations over others. One user recognized clustering of the correlation matrix as particularly helpful. A few team members compared use of the tool to their routine EDA in R and remarked how they were impressed by the large amount of results they can get in a short time. Such comments did not require any changes in the software, but supported confidence in the choices made. Other feedback can be organized into two groups: interface-related and content-related.

Users contributed to the improvement of text information displayed in the app.

If users needed to reread a formulation, this behavior was considered a signal for a change. Moreover, sometimes they said that a given expression was unclear, prompting a change in the interface. For example, the name of a subtab was changed from "Zero and near zero variable predictors" to "Constant and almost constant predictors". The first version was rather technical and users had to pause to understand the meaning. Some widgets also were renamed, e.g. the slider in the *Missing values* tab change its label from "Percent of missing values" to "Show variable with % of missing values over". The text outputs about the substitution of extreme values by `NA` were edited after the first series of usability tests. Over time, some free text information was turned into tabular format.

Visual indication of computationally intensive operations warned users about possible waiting time. Potentially long-running operations can only start be triggered after pressing a button. Initially, the color of the button red in order to signal possible consequences. However, usability tests indicated that using red is not an effective way to communicate the message: most of the participants could not interpret the button color. As a result, this visual solution was rejected.

End users were encouraged to test the software in different ways and their trials resulted in marked improvements in app performance. A continuous variable was assigned as a target variable, and a correlation analysis was undertaken before the removal of constant predictors, to name a few. Such illegitimate actions are currently checked with the `validate()` function and a help message is displayed if an invalid action is attempted.

Probably to the most consistent misunderstanding occurred at the *Modeling* tab. Widgets and controls were positioned on the comparison page in the initial version. It was necessary to choose the settings and switch to the *Model 1* or *Model 2* tabs to see results. Meanwhile, users were expecting to see results on the same page, where the button to train a model was pressed. Fortunately, this shortcoming of the software tool was detected in the first set of usability test sessions. The corrected version of the *Modeling* step was tested during the second series of sessions.

The end users of the software also contributed to the set of offered analytical methods: they suggested the inclusion or exclusion of particular procedures. The initial set of predictive performance measures included neither Brier score nor predictive negative and positive values. Following a colleagues feedback, the list of the resampling methods was extended with a 50/50 holdout. Such split into train and test sets minimizes a potentially harmful effect of erroneous class labels, because they are equally likely to be both sets. The idea to include the number of observations that a particular procedure uses also came from a member of the Data Science team. Furthermore, end users suggested removing random forest importance in the *Variable importance* tab, arguing they would prefer a univariate screening at an early stage of the exploration.

In summary, colleagues' feedback led to improvements of the app, provided guidance in design, and helped to tailor the software to the Data Science team's specific needs. The team's involvement in the app's development has built their trust in the tool and served as an introduction for them to the software. Conducting usability tests iteratively ensured that the severest problems were the first to be fixed, and allowed later testing to fine tune smaller problems. Conversely, group discussions addressed general issues and concerns, and brought new ideas to light.

4 Data analysis

During development, the software tool was tested with multiple real and artificial datasets in order to improve the tool's resilience. Due to confidentiality of SHS VIVEON's client datasets, it is not possible to show an analysis of data used within the company. Instead, this section showcases the developed app using a public dataset similar in theme and complexity to those encountered by the software's end users.

Thematically, the company often deals with data from the e-commerce and telecommunication industries. The firm is tasked with making accurate predictions of shortfall in payment, churn, and promotion campaign effectiveness. Some researchers noted that datasets from open repositories such as UCI are easier to analyze than many real world problems (Japkowicz and Shah 2011, 11). To more accurately reflect one of SHS VIVEON's client datasets, this analysis employs public datasets from machine learning competitions where data was provided directly by enterprises. The chosen dataset is sourced from the Data Mining Cup (2010) and it fulfills the thematic and complexity requirements. Additionally, it contains a business oriented measure to evaluate predictive models.

As described in the competition documentation, a media dealer sells traditional books, audio- and e-books, films, music, and other items. The company's goal is to maximize revenue by improving the effectiveness of one of its promotional activities and customer loyalty measures: vouchers that offer a discount on a second order. The characteristics from the first purchase should be used to identify those customers who would not re-purchase without an incentive. These customers would then be sent discount vouchers.

For the purpose of this demonstration, the media dealer is imagined to be one of SHS VIVEON's client. The first dataset is provided by the client and a member of the Data Science team builds an impression about the information contained in the data, checks the data quality, evaluates the predictive power of attributes, and thinks of ways to improve the predictability of the second order. Appendix B contains a list of the reports of all tabs, metadata, and the extended dataset that can be found on the enclosed CD.

4.1 Initial assessment of the data

The first step is uploading the dataset. The use of the default column separator, a comma, resulted in reading data as one row. The second option of this setting, a semicolon, successfully read in the data. According to information in the *Import summary* tab, the dataset contains 32,428 observations and 38 variables.

The metadata provided by the Data Mining Cup organizers is reproduced in Table 4. The target variable, `target90`, indicates whether a customer made a new order within 90 days. The attributes characterize the customer (e.g. salutation and email domain), the purchase (e.g. payment and delivery type, shipment weight, number and type of purchased items), and any associated promotional activities (e.g. newsletter, redeemed voucher, advertising code) for each observation. The meaning of some variables, such as `advertisingdatacode` and `model`, is unclear. The meaning of number of remitted, canceled and used items also raises questions. The provided documentation does not mention any specific encoding for missing values and empty fields are marked as `NA` as a result of the import.

Column name	Type	Description
`customernumber`	nominal	Unique customer number
`date`	date	Date of first order
`salutation`	nominal	Salutation: 0 = Ms.; 1 = Mr.; 2 = Company
`title`	nominal	Title: 0 = Not available; 1 = Available
`domain`	nominal	Email provider domain: 0 = aol.com; 1 = arcor.de; 2 = freenet.de; 3 = gmail.com; 4 = gmx.de; 5 = hotmail.de; 6 = online.de; 7 = onlinehome.de; 8 = t-online.de; 9 = web.de; 10 = yahoo.com; 11 = yahoo.de; 12 = others
`datecreated`	date	Date account opened
`newsletter`	nominal	Newsletter subscribed: 0 = No; 1 = Yes
`model`	nominal	Model 1; 2; 3
`paymenttype`	nominal	Payment type: 0 = Payment on invoice; 1 = Cash payment; 2 = Transfer from current account; 3 = Transfer from credit card
`deliverytype`	nominal	Delivery type: 0 = Dispatch; 1 = Collection
`invoicepostcode`	nominal	Invoice address postcode
`delivpostcode`	nominal	Delivery address postcode
`voucher`	nominal	Voucher redeemed: 0 = No; 1 = Yes
`advertisingdatacode`	nominal	Advertising data code
`case`	ordinal	Value of goods: 1 = Low; 5 = High
`numberitems`	numeric	Number of ordered items

Column name	Type	Description
`gift`	nominal	Gift option: 0 = No; 1 = Yes
`entry`	nominal	Entry into the shop: 0 = Shop; 1 = Partner
`points`	nominal	Points redeemed: 0 = No; 1 = Yes
`shippingcosts`	nominal	Shipping costs incurred: 0 = No; 1 = Yes
`deliverydatepromised`	date	Delivery date (promised)
`deliverydatereal`	date	Delivery date (real)
`weight`	numeric	Shipment weight
`remi`	numeric	Number of remitted items
`cancel`	numeric	Number of canceled items
`used`	numeric	Number of used items
`w0`	numeric	Number of bound books ordered
`w1`	numeric	Number of paperbacks ordered
`w2`	numeric	Number of school books ordered
`w3`	numeric	Number of ebooks ordered
`w4`	numeric	Number of audio books ordered
`w5`	numeric	Number of audio books ordered (download)
`w6`	numeric	Number of films ordered
`w7`	numeric	Number of musical items ordered
`w8`	numeric	Number of hardware items ordered
`w9`	numeric	Number of imported items ordered
`w10`	numeric	Number of other items ordered
`target90`	nominal	Reorder within 90 days: 0 = No; 1 = Yes

Table 4: Description of variables from the Data Mining Cup (2010) dataset

As soon as the response variable is chosen by the corresponding widget, its distribution is displayed in the *Import summary* tab (Figure 19). The fraction of reorders in the data amounts to 18.7%, which either corresponds to the actual number of recurring orders or is a characteristic of the provided sample of data. In any case, balancing the dataset at the step of building classifiers can be beneficial.

The dataset consist of first purchases by each customer and the number of records coincides with the number of unique customers. Consequently, the `customernumber` is not useful for prediction and it is excluded via the *Excluded variable(s)* widget.

4.1.1 Verification of variable types

A large fraction of attributes are nominal or numeric, a few are dates, and one is ordinal. All columns representing time were correctly identified as dates and dichotomous variables were detected as nominal (`factor`). A comparison of the metadata with the summary table in the app demonstrated the need to correct variable types

of some features. Many nominal variables are coded by integers and were wrongly imported as such. Ordering the summary table by the number of unique values speeds up the detection of further nominal variables: `salutation`, `model`, and `paymenttype`. Moreover, `domain`, `invoicepostcode`, `delivepostcode`, and `advertisingdatacode` were cast as factors. The attribute `w8` is numeric by nature, but was detected as a factor, because it contains only two distinct values, 0 and 1. At the same time, it can be interpreted as a flag variable, indicating whether any hardware items were ordered. Nevertheless, to maintain consistency among the different types of ordered items, `w8` is cast as an integer. Assignment of the correct variable types at the very beginning of the analysis ensures that methods are applied to a correct set of variables, e.g. payment types that are encoded by integers are not used in PCA.

The variable `case` is ordinal and has values ranging from 1 to 5, representing a low to high for the value of goods. As the software does not support ordinal variables, it should be treated either as a higher interval level of measurement or downgraded to a nominal one. Intrinsically, value is a continuous concept so making the case a higher interval could be justified. On the other hand, downgrading it could be more advantageous because distances between categories are not meaningful and the number of distinct values is low. Luckily, the app allows users to look at this variable from both perspectives.

4.1.2 Exclusion of attributes with little variability

The summary table in the *Import summary* tab indicates that many attributes have numerous repetitive values. The variable `points` is a constant, meaning that no customer in the training set redeemed points. Although no further information is provided, this attribute may refer to a customer loyalty program. If any points are gained from purchases, then zero values are plausible for the initial orders. In any case, this constant attribute is useless for the prediction of reorder. Over 99% of customers do not have or have chosen not to fill in their title information. Although presence of this field in the order form may be important from a customer relations perspective, a variable consisting of almost all zeros in unlikely to be a helpful predictor. The variable `gift` describing the gift option also contains almost no non-zero values. For the above reasons, these three attributes are excluded from further analysis.

Among the numeric attributes, several of them contain very little usable information. The summary table in the *Import summary* tab has a column "Percent of most frequent value" that shows how frequently the most commonly occurring value appears in the dataset for a given attribute. Many numeric attributes contain data that is over 95% homogeneous. These variables are integer-valued and mostly have a couple of dozens unique values. In the most extreme case, the number of hardware items ordered (`w8`) includes only two distinct values, with the most frequent one amounting to 99.98%. Based on the summary table, we can assume that only a few hundreds of customers

made any hardware purchase at all. Such a lack of diversity within the data of an attribute is unlikely to produce predictive results. The matrix plot (Figure 6) reveals that the columns corresponding to these variables are almost white, indicating that they contain a high percentage of zeros. The analysis of the dataset can be complicated by the sparsity of this part of the data matrix.

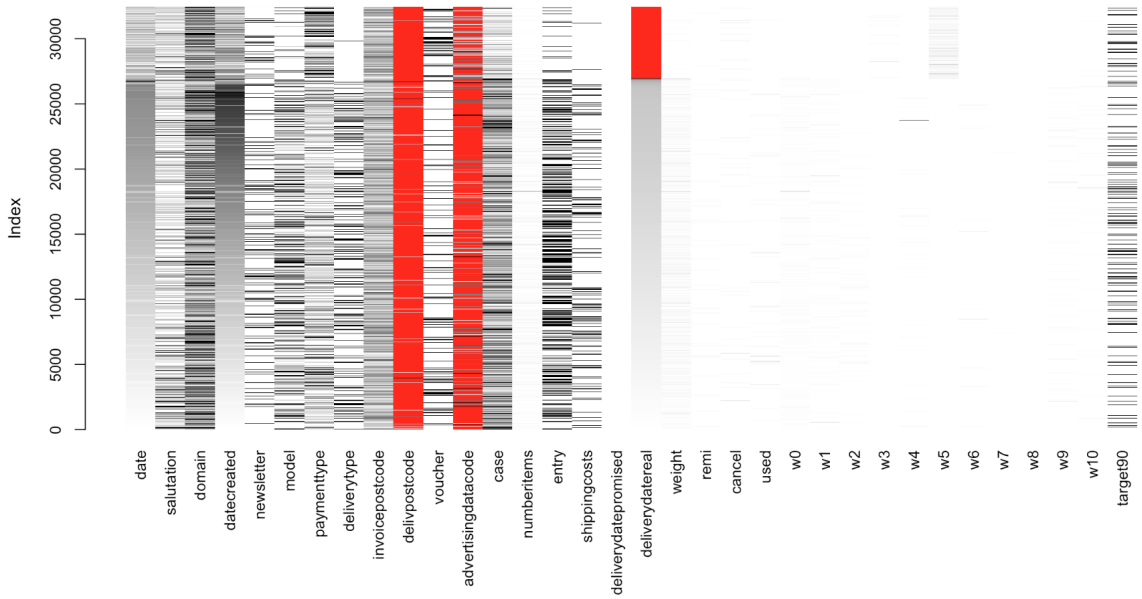


Figure 6: Matrix plot sorted by `deliverydatereal` attribute

The number of distinct values and the percentage of the most frequent value in the summary table reveal potential problems with attributes of all variable types. A more elaborate detection of almost constant predictors among the numeric variables combines the percentage of unique values with the frequency ratio. The default parameters of these two settings suggest a near-zero variance for `used`, `w3`, `w4`, `w6`, `w7`, `w8`, `cancel`, `w10`, and `remi`. For example, the frequency ratio of `remi` is 32.82 and the percentage of unique values is 0.04 (see the complete table in Figure 23). Because the main focus of an analysis is the data's potential predictive power, it is important to take into account variable importance and the distribution of these variables on two levels of the target variable. They are ranked quite low in variable importance and the parallel box plots do not show large differences in distributions. Given these points, they are excluded from further analysis. These variables gave information about which different kinds of media content, e.g. films and ebooks, were purchased by the customer, and such information may yield some predictive power. Therefore, Section 4.5 considers feature construction in order to include information about different types of ordered items.

4.1.3 Exploration of location attributes

The competition documentation does not mention where the customers of the media dealer are located. Because of this, metadata and postal code attributes were analyzed

in order to derive location information. This may be relevant and interesting because the place of residence may embody some information about promotional activities, the company's position in the media market, and potential cooperation with local institutions (e.g. schools) in different areas. In other words, the expectation is not that customers' propensity to reorder differs due to location itself, but rather that the place of residence may act as a proxy for unobserved factors.

The metadata indicates that 9 out of 11 email address domains contain the extension .de (see Table 4). Based on this observation, one can assume that those customers live in Germany. While the postal code attributes are imported as type `character`, coding them as nominal variables allows to inspect different values. The column "Levels" in the summary table shows that the invoice postal code consists of two-digits numbers and the delivery postal codes are either zero, or a two-digit numbers or one of two strings, "Nl" and "EN" (see Figure 7). Without having further details, the strings may mean the Netherlands and England, while numbers may correspond to the first two digits of the 5-digit postal codes in Germany or maybe some encoding due to anonymization and privacy issues. If the German post codes hypothesis is true, the single and double zero factors raise a question, as such codes do not exist. Because there are nearly one hundred different categories codes offer an extremely high amount of granularity, and unlikely to have generalizable predictive power. Assuming that, aside from a few exceptions, the location attributes contain the first two digits of the postal codes it would be possible to bin these codes so that they correspond to the Germany federal states and explore what predictive power this may yield.

Variable	Variable type	N distinct values	Pct most frequent value	Min (character)	Max (character)	Sample values	Levels
delivpostcode	factor	101	3.52	0	Nl	NA, NA, NA, NA, NA	0, 00, 01, 02, 03, 04, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, EN, Nl
invoicepostcode	factor	97	3.84	00	99	58, 34, 01, 51, 25	00, 01, 02, 03, 04, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99

Figure 7: Information about location attributes from the summary table in the *Import summary* tab

The location attributes demonstrate that the dataset contains some nominal variables with a large number of levels (up to a hundred). There is a trade-off between the volume of available data and algorithm's ability to recognize patterns in a given cat-

egory. Moreover, if data for each category is scarce, this may detect some artifacts of a sample. Considering the number of observations, the predictive power of attributes may benefit from aggregation. Such preprocessing is outside of the scope of this initial analysis. After exploring these attributes in more depth, it was decided to exclude the invoice and delivery postal codes from further analysis.

4.1.4 Exploration of time attributes

The data presents a year of records, as marked by a range of order dates from April 1, 2008 to March 31, 2009. The lowest values of all four time variables (`date`, `datecreated`, `deliverydatepromised`, `deliverydatereal`) correspond to year 2008. The summary table indicates that the latest promised delivery date is 4746-11-26 (supposedly November 26, 4746). The large range of values of `deliverydatepromised` is reflected in the matrix plot by much whiter coloring than the other date columns (see Figure 6). The maximum value for the actual delivery date is rather high, 2009-12-30, i.e. this customer waited at least nine months for a delivery. Although not impossible, such a time period seems rather unusual for media content and may require a follow-up with domain experts from the media dealer.

Time is presented in the dataset by fixed date fields (e.g. when an account was created). This is a natural way to store information in a database and may be preferable to variables derived from them (i.e. time since an account was created). However, dates are not recognized by the learning algorithms, so it is recommended to derive some numeric or categorical variables from the date fields. As the software does not offer this functionality, date fields are excluded from subsequent analysis after their exploration and a sanity check.

4.1.5 Inspection of missing values

A failure to deal with missing values would result in under 1% of the original number of complete rows being usable for analysis (see Figure 8). The dataset contains three attributes with unobserved measurements, their percentage ranging from roughly 17% to over 95%.

The delivery address postcode contains the highest percentage of missing values. There are several reasons why this may be the case, which are visible from sorting of the matrix plot by delivery type. First, all orders with `delivery-type = 1` ("collection") contain no delivery address. Unambiguous interpretation of the situation would require details on the meaning of two delivery types, "dispatch" and "collection". Probably, the latter means that the purchase was picked up

There are 207 (approximately 0.64%) complete observations.

	% of missing values
delivpostcode	95.7074
advertisingdatacode	79.8847
deliverydatereal	16.8743

Figure 8: Numeric summary from the *Missing values* tab

at an offline store. If this idea is true, the field can be imputed by the postal code of the shop, because this implies that the customer indeed located nearby. In fact, sorting the matrix plot by payment type (the report can be found on the enclosed CD) demonstrated that collected orders were paid in cash, which supports this hypothesis. Second, a customer may leave the delivery address in the order form blank if both the invoice and delivery addresses coincide. Assuming it is the case, the missing values for the orders with delivery type "dispatch" could be imputed by copying the postal code from the invoice attribute.

The missing values in the advertising code may indicate "none", which can be introduced as an additional category. Overall, there are over 40 different codes and details about abbreviations, e.g. AP, BR, BQ, may help to build aggregate categories for generalizable performance. The percentage of missing values in this attribute is high, almost 80%. Omitting observations with a missing advertising code would potentially strongly bias a dataset, especially if the hypothesis that missing values corresponding to no advertising code is correct. Consequently, this attribute can be excluded or missing values can be substituted with an additional category "missing". Despite the potential predictive power of this promotion-related variable, its high number of levels favored its exclusion.

An inspection of the matrix plot sorted by actual delivery date shows a distinctive pattern in some other variables (see Figure 6), e.g. date of account opening and of the first order. Potentially more revealing is its relation to `w5`, the number of downloadable audiobooks. Importantly, there are two variables describing audiobooks and, although it is not completely clear, `w4` may account for CD sales of this genre while `w5` for recordings available for download. Assuming this hypothesis is true, the delivery date can be estimated by the order date as downloads are immediately available, regardless of the actual download time.

In summary, the missing values in all three variables seem to have plausible explanations. Additional information about the data collection process has the potential to impute values. The inspection of the summary table in the *Import summary* tab and the matrix plot in the *Missing values* tab increased the understanding of the data, helped to resolve some ambiguities present in the provided metadata, and provided a context for the missing values. It also pointed toward other potentially helpful preprocessing steps, e.g. binning of postal codes. Moreover, this inspection highlighted the usefulness of excluding some of the attributes.

4.1.6 Inspection of extreme values

The two methods for detection of extreme values in the *Preprocessing* tab produce very different results (see Figure 9). The mean-based rule when run at its default settings would decrease the number of complete cases by 2600 (about 8%). In contrast, the median-based rule would reduce the entire sample by two thirds. This indicates

some unusual structures within the data. For one, the high percentage of repeated values plays a role. The decision rule based on medians would drastically reduce the number of complete cases due to high percentage of the most frequent value. To illustrate, attributes describing the number of specific items (`w1`, `w2`, `w5`, `w9`) are characterized by both quartiles coinciding with the median of 0. In other words, over 75% (and often over 90%) of these variables have zero values and variability is almost absent. Therefore any t would remove all non-zero values.

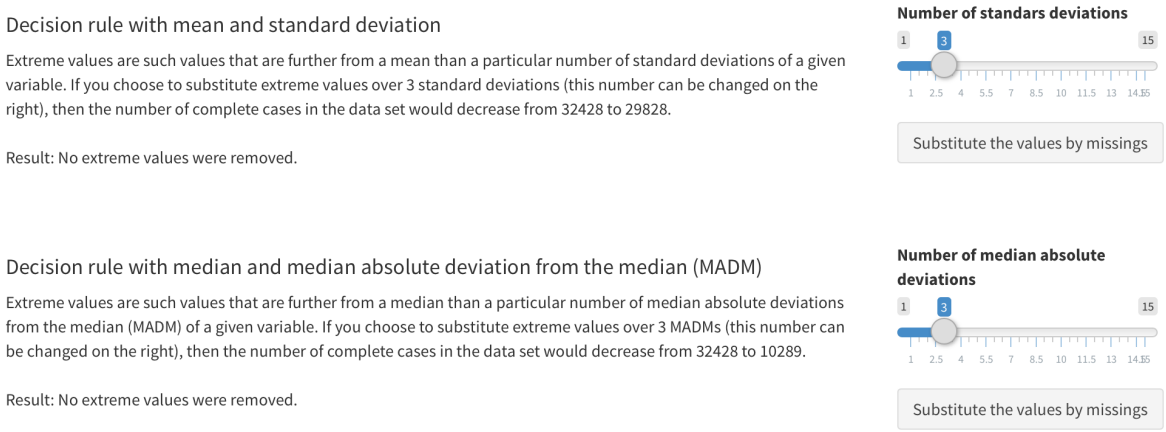


Figure 9: Output of the *Preprocessing* > *Extreme values* tab

A long right tailed distribution of almost all numeric attributes demonstrate a rarity of high values. The right skew is evident by numerous points outside of upper box plots whiskers (1.5 IQR) and no unusually low values. Moreover, there is often no lower whisker and both quartiles coincide with the median. This demonstrates a concentration of the mass of the values in a single number of a narrow range of values. Furthermore, distributions of this shape make kernel density plots a poor graphical display.

An examination of extreme values in the box plots reveals what are most likely erroneous entries in some of the ordered items. According to the initial understanding of the metadata, `numberitems` is the total number of ordered items, which is split into categories, e.g. paperbacks, films, etc. However, maximum of this variable is 50 and other high values are under 30 items, while maximum of particular class attains 84 (`w1`), 90 (`w2`) and 99 (`w0`). This either indicates errors in number of items in categories or the meaning of `numberitems` is different.

Because these extreme values are likely to be the result of errors, it would be best to substitute them with missing values before further analysis. Moreover, high single values in `numberitems`, `weight` and `w9` seems atypical for the data at hand. A goal to remove only extremely high values in selected variables is achieved by first temporarily excluding other numeric variables and then applying a substitution for values beyond the highest number of standard deviations available in the software, 15. This unusually

high number of standard deviations takes into account that, due to little variability, the measure of spread is also very small. This operation results in reducing the number of complete cases by 80 observations. A subsequent review of the box plots demonstrates that the upper bound of different types of items has been reduced to under 50.

The weight attribute also contains some unusually high values. The upper bound is clearly an extreme value, because other high values are around 10,000. The minimum shipment weight is zero, which is realistic for orders with downloadable items only. According to the matrix plot, low values of shipment weight plausibly relate to a substantial number of downloadable audiobooks as well as ebooks. Although the metadata contains no measurement units for this variable, its spread suggests measurement in grams.

4.2 Exploring relationships between variables

4.2.1 Relationships between an attribute and the target variable

The association measures based on the reduction of error in prediction (Goodman and Kruskal λ and η^2) are zero for all attributes. Values of ROC AUC in the variable importance are low, from 0.5 to 0.54 (Figure 10). This means that even the higher ranked attributes are of little help in predicting reorders. In addition to this variable ranking, the relationships between single attributes and the target variable are examined graphically and with the help of association measures. First categorical (Figure 11) and then numeric variable (Figure 12) are considered.

Vouchers were used in approximately 15-20% of initial orders, which has almost no association with reorder: the attribute `voucher` is ranked low by the variable importance and the values of Cramér's V is 0.03. Intuitively, experiencing the benefit of this promotion has the potential of increasing the willingness to reorder. According to the matrix plot, turning in a voucher seems to couple with some advertising code. Overall, the variety of advertising codes (over 40 kinds) and their relationships with other attributes suggest that there are multiple promotional campaigns going on simultaneously and customers' decision to repurchase may be influenced by incentives other than a voucher. Consequently, the dataset is potentially missing important predictors.

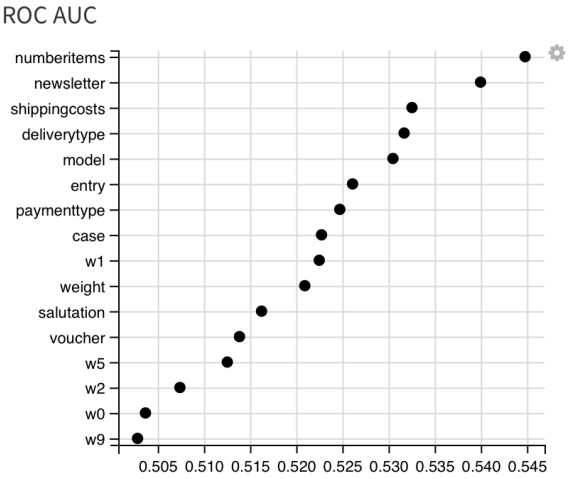


Figure 10: Attribute ranking from the *Variable importance* tab

A customer's subscription to the media distributor's newsletter is a rather strong pre-

dictor (ranked second by the variable importance, Cramér's V is 0.08) and the mosaic plot demonstrates a higher reorder rate by those who receive this type of communication (see a plot for the `newsletter` attribute in Figure 11). Possibly, newsletters have relevant content or special offers and somehow motivated customers to repurchase. However, a confounding variable, such as general interest in the analyzed media dealer, may determine willingness to subscribe and to reorder. While the merchant can be interested in the effectiveness of their promotional efforts, the current task of the data analysis is to improve prediction.

Customers who paid in cash for the initial payment repurchased more often in comparison to those who used other types of payment (see a plot for `paymenttype` in Figure 11). Distribution of the remaining payment types (namely payment on invoice, transfer from current account, and from credit card) in relation to the target are nearly the same. This trend is not necessarily connected to the payment types themselves, as the collected items were all paid for in cash and the percentage of repurchases among the customers who collected orders is higher. Furthermore, both variables, `paymenttype` and `deliverytype`, were only weakly associated with the target variable (Cramér's V is 0.06 and ROC AUC of 0.53). In summary, these attributes seem to encode similar information in relation to the target variable.

The absolute majority of the customers (about 80%) did not incur any shipping costs with their order and, indeed, this fact is associated with a higher reorder rate (Cramér's V is 0.07). Some of the customers without any shipping costs picked up their orders at a local shop, while others with free delivery, which is *ceteris paribus* an advantage. No information about the conditions associated with free delivery are available, but according to the matrix plot it seems to weakly correlate to a lower number of items and somehow rather low value of goods (`case = 2`).

Out of common sense, the email address domain information would not have a relationship with the reorder rate and, indeed, the estimate of association is weak (Cramér's V is 0.02). Moreover, some levels of this nominal variable contain few observations and a category "others" indicates that there were even less frequent domains. For these reasons, this attribute is excluded from further analysis. Nevertheless, the media dealer can consider web-marketing at different websites and this information can be relevant.

The ordinal attribute `case` should be treated either as nominal or numeric in the software. Sorting of the matrix plot by this variable demonstrated that the five categories of goods' value are fairly evenly distributed with a somewhat lower fraction of the highest rank. There is no absolutely clear co-occurrence with other variables, but percentage in vouchers is relatively high in the lowest value category. The description of `case` does not specify how it was derived. In particular, whether it corresponds to the cost of goods sold, the purchase price before or after discounts, etc. If the

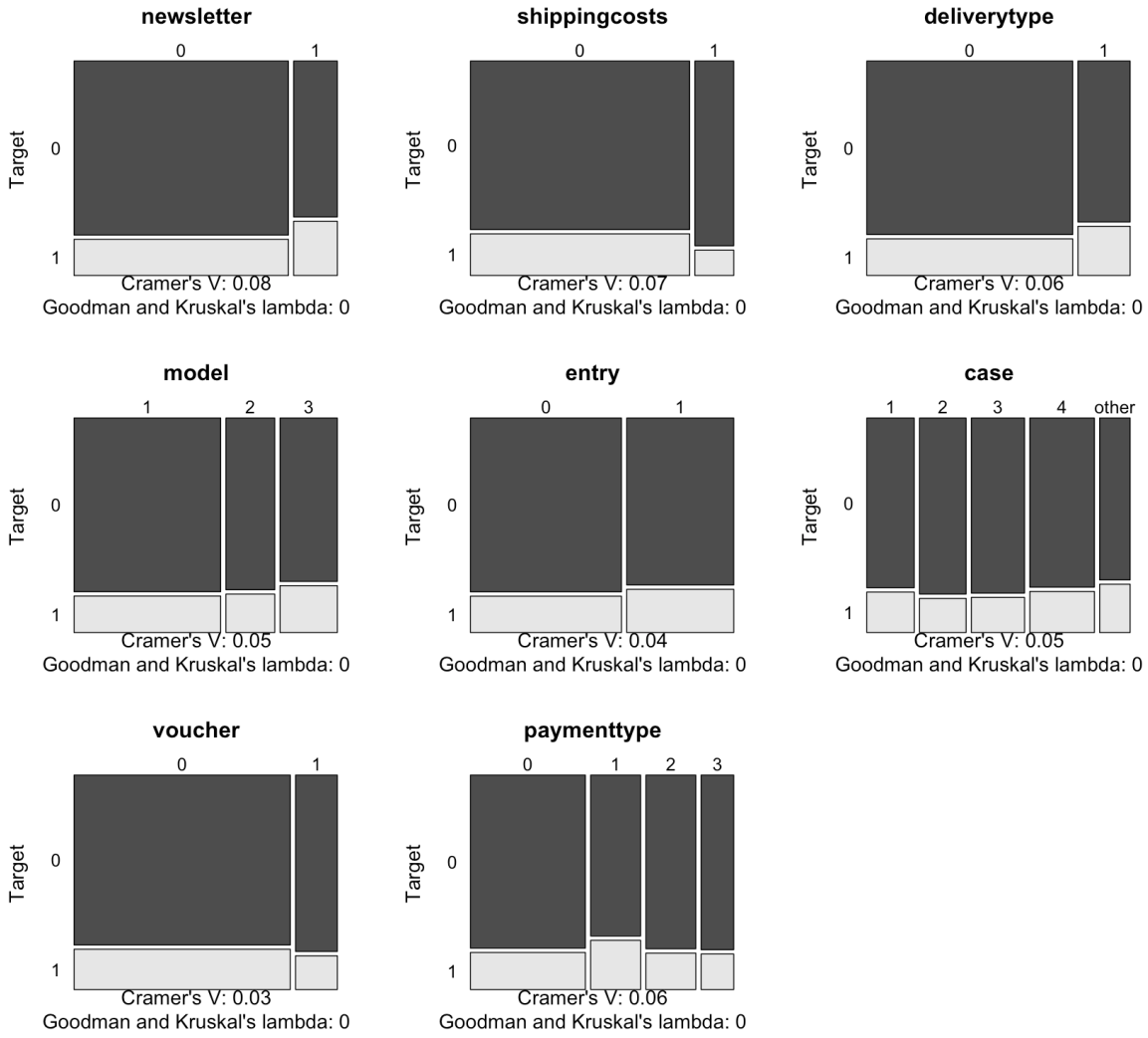


Figure 11: Mosaic plots from the *Categorical variables* tab

value of `case` is calculated after applying a discount, then the presence of a higher percentage of vouchers in the lowest values is not surprising. If `case` is treated as numeric, the box plot is symmetric on the both levels of the target variable (Figure 12). If the `case` attribute is treated as nominal, the clustered bar plot demonstrates similar distributions at two levels of the target. Association is not strong (Cramér's V is 0.05), but the mosaic plot shows a slightly higher percentage of repeated purchases in the highest value of goods of the initial order (Figure 11). This may be connected to a higher interest in media items in general. The continuous nature of the value of goods favors the quantitative treatment of the attribute, so it is treated as integer for the consequent analysis.

The parallel box plots of the numeric variables have rather similar distributions on both levels of the target variable (Figure 12). Most of the attributes demonstrate differences only in the values beyond the third quartile. Higher number of items and weight are observed for customers, who did not repurchase. Such large orders are overall atypical and may correspond to one time purchases.

The `numberitems` has the highest rank in the variable importance, but the value of ROC AUC is quite low, 0.54. The customers who did not reorder within 90 days tend to have purchased fewer items in their initial order, in particular, minimum, median and first quartile coincide at value of 1. The median of another subgroup is 2. No information about any influence of the second purchase is available, but maybe such customers regularly consume more media content on average and returned to buy more.

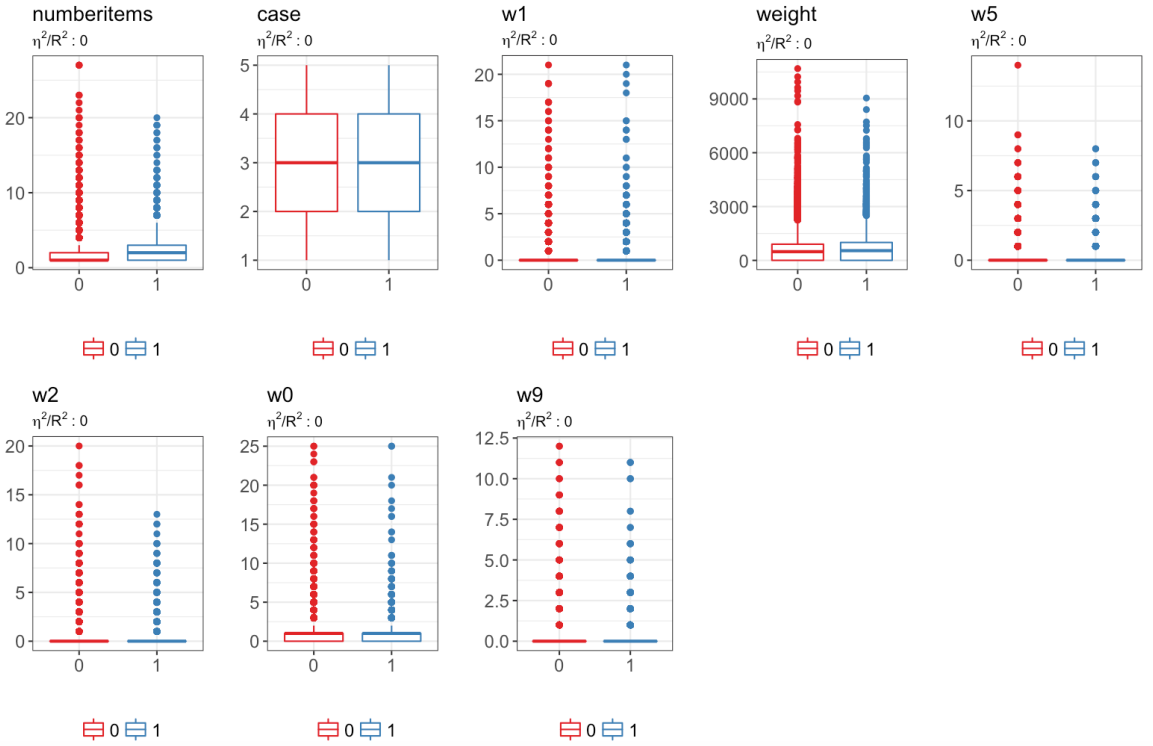


Figure 12: Parallel box plots from the *Numeric variables* tab

4.2.2 Relationships between attributes

Association between attributes is explored with the matrix plot, correlation plots and PCA. Section 4.1 described relations between variables with missing values and all others which generated ideas for imputation. Arranging the matrix plot by various variables helped to clarify their meaning and uncover co-occurrences.

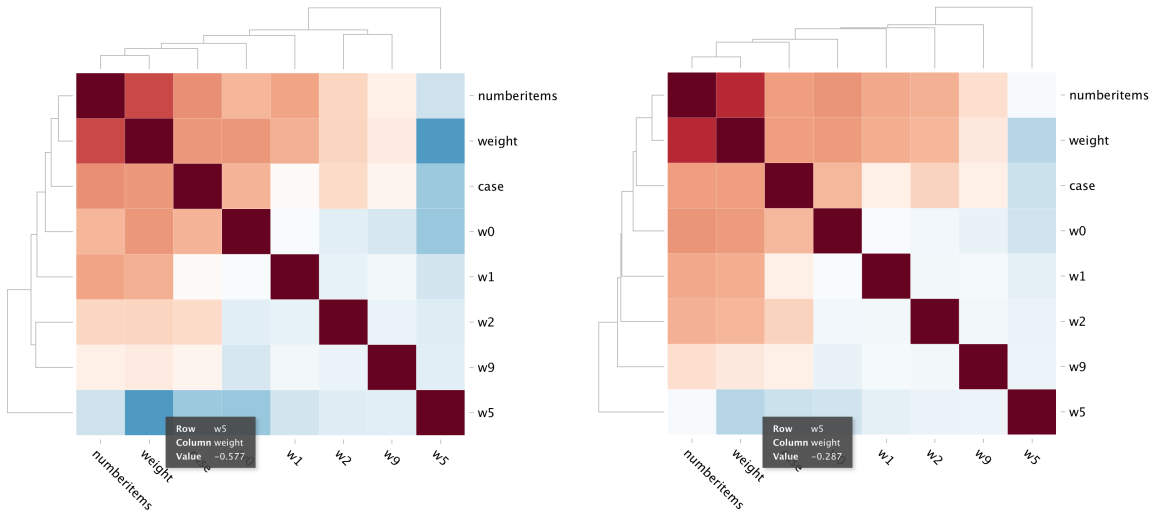
As noted in Table 4, the metadata only characterizes `model` as 1, 2 and 3. Sorting the matrix plot by this variable revealed its relation to how customers entered the shop: all orders in model 2 and the absolute majority in model 3 are made through a partner (`enty = 1`). At the same time, model 1 corresponds to the entries through the shop. This means these two variables, `model` and `entry`, contain almost the same information.

The simultaneous inspection of a few other variables sheds light on the delivery type and entry attributes. The type "collection" (`deliverytype = 1`) amounts to approximately a fifth of all orders and is associated with no shipping costs and cash payments.

Plausibly, dispatched orders are paid on invoice or via a transfer. Additionally, a large proportion of collected orders occurred though a partner. Sorting the matrix plot by advertising code also points out that its presence almost never co-occurs with an order through a partner. Possibly, a partner company offers pick up points for orders and allows cash payments, but is not much integrated in promotional activities of the media dealer.

The presence of many repetitive values and the shape of distributions of the numeric attributes make correlation analysis and PCA not fully reliable. Nevertheless, the strongest positive correlation is between the number of items and weight (Pearson correlation is 0.776 and Spearman is 0.664, see Figure 24), which is plausible considering the products' nature. Highly correlated attributes are likely to encode similar information in relation to the response variable (Myatt and Johnson 2014, 143). This may motivate end users to exclude one of the attributes that have a strong relationship with each other. However, the strength of the association present in the dataset was not considered high enough to justify the exclusion of any variables.

Overall, sample Spearman rank correlation coefficients are quite similar to the product-moment correlations (compare Figures 13a and 13b). A rather high difference between two measures is observed for the pair of shipment weight and the number of downloaded audio books. The Spearman coefficient is -0.557 while Pearson one is -0.284. Such a difference may indicate curvature, extreme values or other structures in the data. Moreover, the number of downloaded audio books is negatively correlated with all the other numeric attributes. Intuitively, this type of media content is quite distinct and negative correlation is justifiable.



(a) Spearman (b) Pearson

Figure 13: Correlation plots of attributes excluding nearly constant predictors

Weak correlations pose a challenge for linear dimensionality reduction. There is little variability in the numeric variable and, intuitively, PCA is not a helpful technique for

the data. All numeric attributes that remain, `numberitems`, `weight`, `w0`, `w1`, `w2`, `w5`, `w9`, and `case` correspond to a purchase itself and PCA offers a way to look at their correlation structure. The eigenvalues drop to approximately 1 after the first component. The scree plot criterion suggests only one component (Figure 26), which explains 35% of the variance.

4.3 Understanding groups

The company works to make predictions on the customer level and uncovering groups of customers without relation to their reorder decision has a potential to increase the media dealer's understanding of their clients. Each row of the data represents first orders and, therefore, each observation corresponds to a customer. The *Clustering* tab offers four data-driven grouping of individuals, which are, however, bound by the implemented measures of proximity. Having no prior knowledge, Step 1 for choosing a number of clusters was performed in every clustering method. The maximum number of groups is limited by the capacity to interpret clusters and was set to eight.

Internal validity criterion available for K-means resulted in diverse recommendations. Silhouette width criteria and Calinski-Harabasz pseudo F-statistic suggest two clusters, Davies-Bouldin's index – one, and the gap statistic – five. The average silhouette width exceeds 0.5 for two- and five-cluster solutions which corresponds to reasonable structure (Kaufman and Rousseeuw 1990, 88). The enclosed CD contains reports for K-means with both two and five clusters.

The two-cluster solution split all customers approximately in half. The second group is characterized by higher value of goods and larger orders in general (attribute `numberitems`) and also in all subcategories but downloadable audiobooks, where higher values for the first cluster are observed. This suggests a distinction of customers tastes for physical items and downloadable content. The five-cluster solution clearly divides customers into groups corresponding to five categories of values of goods of approximately equal size. The first cluster is characterized by the highers value of `case` and overall tend to have purchased larger number of items. On the contrary, the second cluster is described by the lowest values of goods and higher number of downloaded audiobook (third quartile has a value of one in comparison to zero in other clusters).

Both internal criterion for CLARA method suggest six clusters with relatively high value of the silhouette coefficient (interpreted as strong structure by Kaufman and Rousseeuw, 1990). The CLARA method uses Manhattan distance, which is different from K-means, but overall the solutions of these two clustering methods based on the numeric variables are similar. A division by value of goods is also detected by the CLARA method. The two smallest clusters (four and five) correspond to the lowest value and are different in the kind of media content purchased, physical books and downloadable items. The last group (about 12%) is characterized by purchases of high

quantities of bound, paperback and school books and the highest values of goods. The media dealer may be interested in attempting to increase its share of such customers, due to this potential to generate high revenues.

The average silhouette width criterion suggests two PAM clusters with three clusters also having a similar value of this statistic. However, in absolute value they are quite poor, around 0.3, which corresponds to a weak structure (Kaufman and Rousseeuw, 1990). The two cluster solution divided the sample in roughly equal groups. According to this clustering technique, customers most substantially differ in regard to their entry: shop or partner (which also implies differences in model, because these attributes contain similar information; see Section 4.2).

A dendrogram for the single linkage (a plot on the right in Figure 14) indicates that a so called chaining problem is apparent for the dataset (as expressed by Gower dissimilarities). Average linkage indicates small groups of observations (right part of the dendrogram for the average linkage) that were merged with others at a rather late stage of the agglomeration process. These customers seem to constitute a distinct group and demonstrate that clustering has detection of unusual observations as its by-product. The complete linkage seems to have relatively large decreases in dissimilarity in the first few splits. The silhouette width criterion suggests three clusters for this agglomeration method, which are also clearly seen from the dendrogram. As a result, Step 2 of the hierarchical clustering was performed with complete linkage and three clusters.

A description of the clusters in terms of individual variables demonstrates that categorical variables seem to differentiate clusters resulting from the complete linkage better than the numeric ones. The smallest third group (around 20%) consists of customers who collected their orders and paid for them in cash as this delivery type and payment method are not present in other clusters. At the same time, both values of the entry attribute (shop and partner) characterize the third cluster. An inspection of the numeric attributes demonstrates that customers from this group bought no downloadable audiobooks, which is probably connected to the delivery method. The first (around a quarter of customers) and the second clusters (over half) are similar in terms of payment methods used, shipping costs, and use of a voucher. However, they are different in entry: the first cluster is made up almost exclusively of customers who entered the shop though a partner business while the second is made up of customers who entered through the shop itself. As for buying behavior in terms of orders, on average the customers from the first cluster purchased more items than those of other groups. Overall, the hierarchical clustering with the complete linkage separated groups of customers by particular characteristics. However, the reorder pattern is similar in all groups and, consequently, this clustering does not add much to the predictability of repeat purchases.

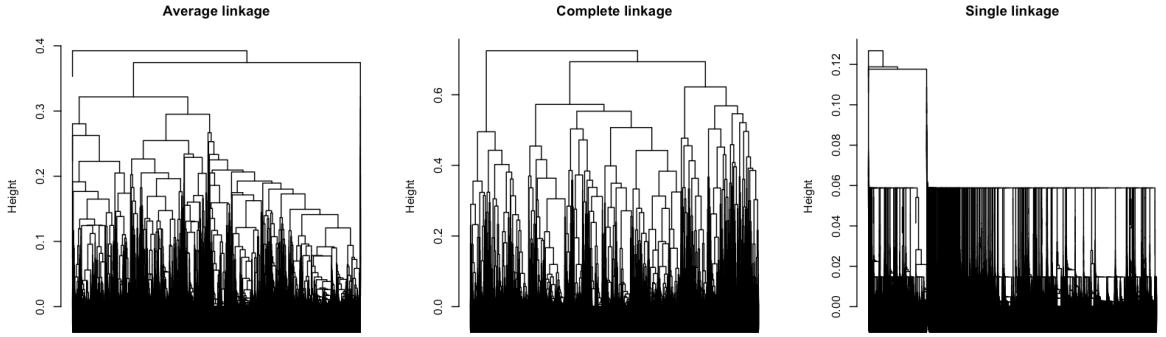


Figure 14: Dendrograms and average silhouette width plots for hierarchical clustering

4.4 Building classification models

An exploration of the relationships between variables showed that the dataset contains some predictive power, but important influences are probably not observed. In other words, a classifier that includes only the features that are present in the dataset cannot account for all factors that influence the decision of a person to make a second order. The task of accurate human behavior prediction based only on a customer's first order at an online store is probably unsolvable.

Nevertheless, a classifier can be useful even if a model is not particularly strong in absolute terms. A business metric developed by a domain expert makes measuring usefulness more straightforward. One possible approach is net revenue gain, after a cost of the model is accounted for. Furthermore, a cost-benefit analysis raises the question of whether further investment in improving a model is worthwhile. The competition documentation includes a cost matrix (Figure 15 shows it reflected in the tool). A voucher gives a discount of €5, therefore sending it to customers that would repurchase without it results in a loss. The media dealer assumes that 10% of those that would not otherwise reorder, would complete a €20 purchase on average. Accounting for the discount, this results in a €1.5 gain. The cost matrix allows an analysis to establish a base performance in the whole dataset of $26{,}377 \cdot 1.5 + 6{,}051 \cdot (-5) = 9{,}310.5$. Consequently, a predictive model resulting in a higher net revenue gain can be considered useful.

Decisions about positive class, resampling and the use of original, down- or upsampled datasets are made before any model is trained. The formulation of the cost matrix suggests keeping `target90 = 1` as a positive class. Positive predictions do not result in sending out a voucher and therefore, zeros are assigned to TP and FP. The FN prediction results in a loss, -5, and 1.5 is set for TN, see Figure 15. The monetary evaluation also encourages the resampling method, 5 or 10-fold cross-validation, because the holdout approach would contain only a fraction of the data as a test set and no direct comparison with base performance in terms of revenue is possible. For the first iteration, a 10-fold cross validation and the original dataset are used.

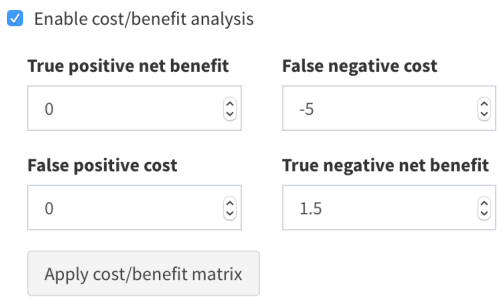


Figure 15: Cost-value matrix

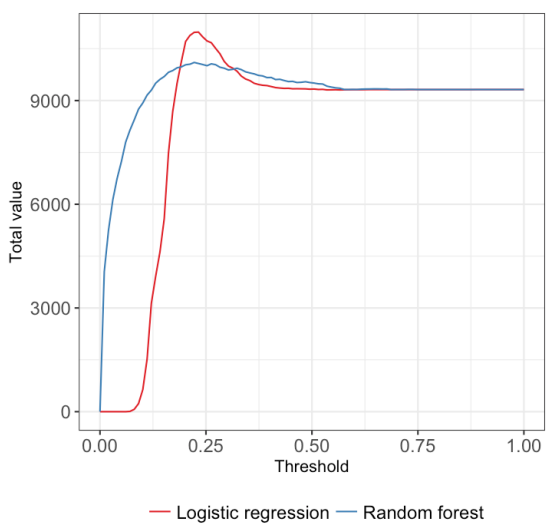
The analysis of attributes decreased the number of fields due to missing values, a high number of factor levels, near-zero variance or format (dates). The variable importance ranking allowed to concentrate on attributes with the strongest predictive power, but no variables with low ROC AUC were excluded solely for this reason. This was done intentionally, because the variable importance represents only a univariate metric. Filtering out the attributes based on the strength of relation to the target variable does not let learners explore the joint effect of variables. If one choose best predictors, e.g. by univariate variable importance, this implies that data indirectly has seen the target and therefor, the estimation of performance is not honest. Unsupervised screening and consequent exclusion of attributes does not have this effect, but supervised screening gives an advantage to the chosen variables (Hastie et al. 2009, 245–247).

All learners outperformed the base performance in terms of net revenue gain of €9,310.5. k-NN and a decision tree offered only a slight increase and, therefore, were not considered for further comparison. Logistic regression and random forest achieved a moderate increase of net revenue by 18% and 8% respectively. The maximum revenue gains are calculated by finding the maximum of total value as a function of a probability threshold, depicted in Figure 16a. The ROC curve of logistic regression is also uniformly above the curve for the random forest (Figure 16b). The chosen cut-offs for two learners are equal to 0.22, i.e. instances with predicted probability above this value are predicted to reorder. Figure 17 demonstrates that logistic regression performed better than a random forest in a variety of metrics.

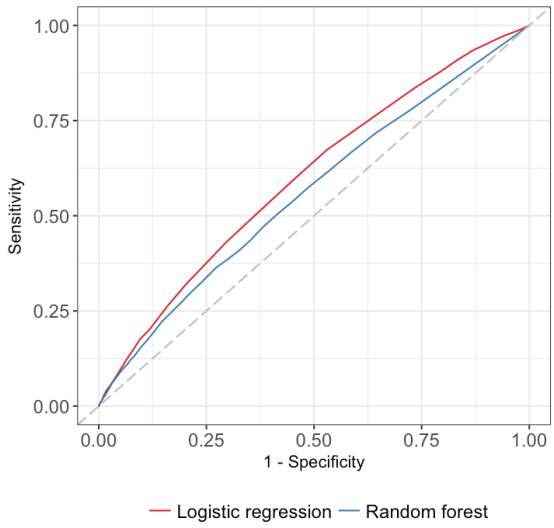
In absolute terms, the performance of classifiers is rather poor. The AUC of the logistic regression is 0.6, which is not much higher than random. The value of Cohen's κ, 0.11, also does not qualify for a reasonable agreement between actual and predicted outcomes. A visualization of predicted probabilities for different true outcomes, reorder or not, confirm strong overlap of their kernel density estimators (Figure 18). Nevertheless, the net revenue increase indicate classifiers' usefulness.

4.5 Conclusions and further analysis

The completed analysis demonstrated that predictive models have a potential to increase revenue. Overall no model can predict probabilities of a second order well. At the same time, in terms of change in revenue, the learners achieved higher values than the base amount of €9,310.5. The best result, an 18% increase, is achieved by logistic regression.



(a) Net revenue gain



(b) ROC curves

Figure 16: Graphical comparison of two learners

	Logistic regression	Random forest
Sensitivity (TPR)	0.301	0.171
Specificity (TNR)	0.813	0.889
Positive predictive value	0.270	0.261
Negative predictive value	0.836	0.824
Accuracy	0.718	0.755
Balanced accuracy	0.557	0.530
Cohen's Kappa	0.110	0.069
AUC	0.601	0.557
Brier score	0.149	0.165
Total value	11042.500	10085.500

Figure 17: Numeric comparison of two learners

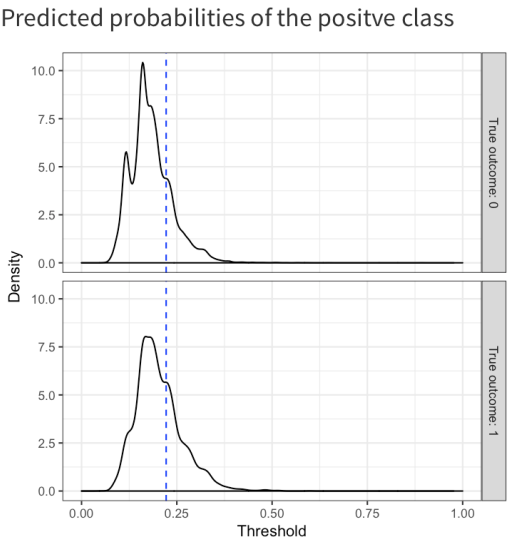


Figure 18: Predicted probabalities of reorders by logistic regression

Clustering methods suggested different numbers of groups and various distinguishing characteristics, e.g. the value of goods or entry into the shop. The choice of the number of clusters was guided by internal validity criterion and the results interpreted on the basis of included variables. An evaluation of the results by domain experts and experimentation with distance measures has the potential to discover even more distinctive groups. The initial analysis demonstrated interpretable results and provided the media dealer with insights and ideas about their customer base.

In the next iteration, different approaches to strengthen the "signal", and consequently increase revenues, can be explored. The first approach would be collaboration with domain expert (e.g. from the media dealer) in order to engineer new features.

Hypotheses about the imputation of missing values require validation from the media dealer. The second approach would involve cleaning the dataset (e.g. dates from year 4746), imputation of missing values and univariate transformations (e.g. aggregation of categories of nominal variables, binning of numeric features). The third approach could be to build derive variable as a function of one or multiple attributes.

The high number of levels in nominal variables poses a challenge for making generalizations about their relationship with the response variable. Therefore, reclassification of such variables has the potential to increase their predictive power. The first approach uses domain knowledge and business logic. To illustrate, street-level addresses can be aggregated into larger geographical ares, e.g. cities, however, this may remove some insightful dependencies. Sometimes location is believed to represent a unique characteristic, having its own predictive power. For example, 50 US states can be transformed into an ordinal variable, describing economic level (Larose and Larose 2015, 42–43).

An described above, the location information from invoice and delivery postal codes can be aggregated to represent the German federal states. A challenge of high number of categories in `advertisingdatacode` may be resolved by turning it into a binary variable: advertising code present or absent.

The second way of reclassifying categories relies directly on the target variable: levels with similar relation to the response variable can be grouped. Section 4.3 describes the fact that the rate of reorder was more or less the same for all electronic payments. Therefore, it may be useful to combine payment on invoice, account transfers, and credit transfers into one class.

As mentioned above, many numeric attributes describing the different kinds of items purchased had been removed from the initial analysis due to their near-zero variance. However, the derivation of new variables may allow them to be incorporated. A few ideas include aggregation (e.g. group physical items and those related to ebooks, downloads), conversion into binary variables (whether this item type was ordered), building a proportion of total number of items, etc. Turning `remi` and `cancel` into binary variables would not solve the issue of high percentage of repetitive values. After the meaning of these variables is verified with the data providers, they may be incorporated into a new variable, characterizing the total number of items ordered (`totalnumberitems = numberitems - remi - cancel`).

Time information represented by dates is a rich source for building new features. In particular, time differences is a natural way to combine information from attributes. A reorder means that a customer returned to the website. A time difference between creating an account and the first order may turn out to be a proxy for a return behavior. If this intuition is correct, the customers who made a pause between making an account and their first order may be more likely to reorder. Nevertheless, the

same behavior may have been due to problems with payment and in such a case the lapse of time between account creation and purchase would be an indication of a negative experience. Leaving aside these speculations, building a numeric variable for the number of days between account registration and first order is a way to use time information. Another idea would be to calculate the discrepancy between the promised and real delivery dates. This can be done with or without the imputation of `deliverydatereal` described above, but definitely after substitution of extreme values in `deliverydatepromissed`. Furthermore, the choice of the merchant to track reorders only within 90 days is arbitrary. It would likely be productive to lengthen the timeframe for logging reorders, while also incorporating more data about the customers and their purchasing habits.

5 Conclusion

Solving any problem with the help of data is an iterative process. The developed tool offers a quick way to make the first iteration for predictive modeling. The current release of the software provides a comprehensive set of core functionalities allowing users to carry out an initial EDA. The software gives analysts a first glimpse of the predictive power of the data, fluidly, without spending excessive effort to execute repetitive programming. Designed for classification tasks, the tool grants the analyst the ability to assess data's suitability for outcome prediction and to concentrate on examining the strongest predictors.

End users access the app through a web-based graphical interface, organized as a series of tabs, each dedicated to a specific aspect of analysis. The software deliverables empower decision makers by providing them with a variety of visual and statistical representations of the data. The results of the analysis are downloadable as .html reports enriched by users comments and as an extended dataset, e.g. with added cluster membership.

The software includes panels dedicated to missing values, variable importance, numeric and categorical variables, clustering, and predictive modeling. Both the author's research and the suggestions of the members of the Data Science team at SHS VIVEON AG contributed to the choice of features for the app. Some rather simple techniques were elected to be included in the app so that users could run the data through these functions in order to gain a better understanding of the types of information they are working with, and to perceive the complexity of the problems they are trying to solve.

User research in form of focus groups and usability tests involved bringing end users into the development process. Colleagues' feedback provided the basis for the improvement of the app, allowing it to be tailored specifically to meet their needs. This feedback confirmed that the interactivity, speed, and ease of use were advantageous for analysts when they used the tool.

The implementation in R enabled flexible development and ensured the possibility of modification of the tool by the members of the Data Science team in the future. Numerous packages mentioned throughout this thesis provided access to a wide variety of descriptive statistics, clustering, and predictive modeling. The tool was turned into an R package and installed on the internal server of the company during its final stage of the project.

Future development of the tool may include both functional and performance improvements, for example, the extension of functionality to quantitative response variable. The inclusion of spacial aspect of data (if present) may also be a desired feature. Performance improvements may include search for R functions that are optimized for large datasets and improving memory-efficiency of the programming code. Higher calculation speed improves user's experience and may encourage the analysts to follow the leads in the data without computational limitations.

Appendices

A Software screenshots

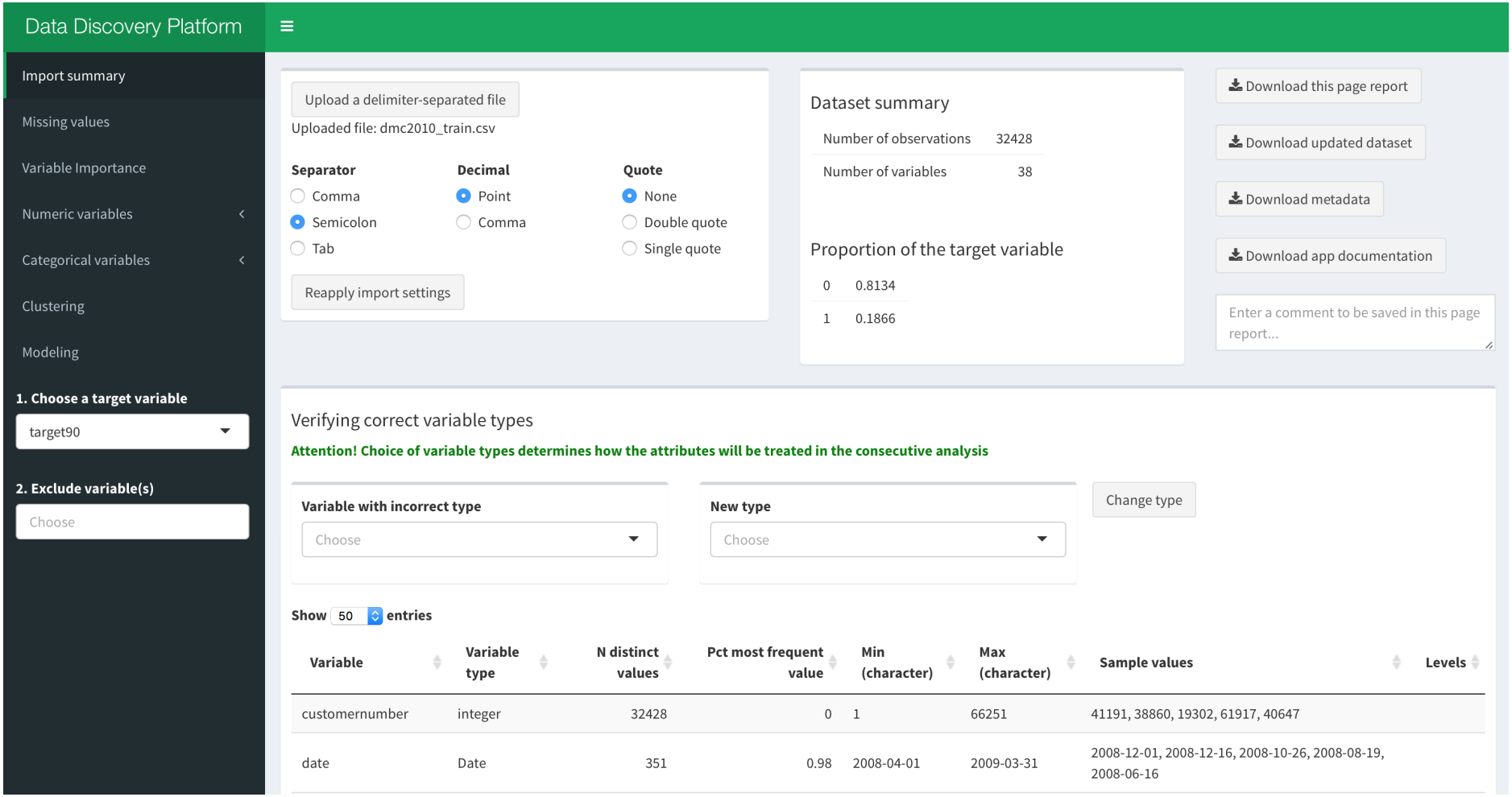


Figure 19: *Import summary* tab

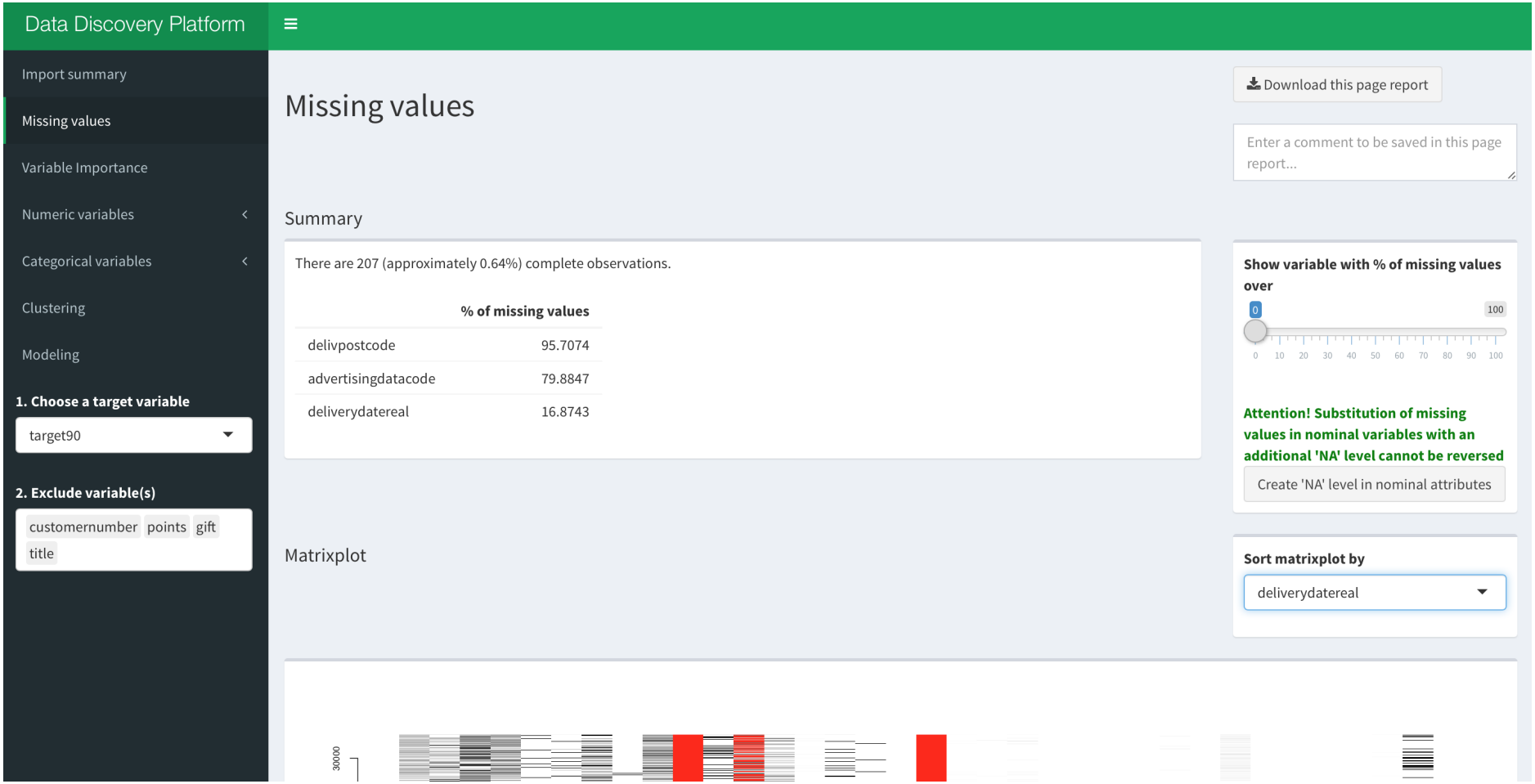


Figure 20: *Missing values* tab

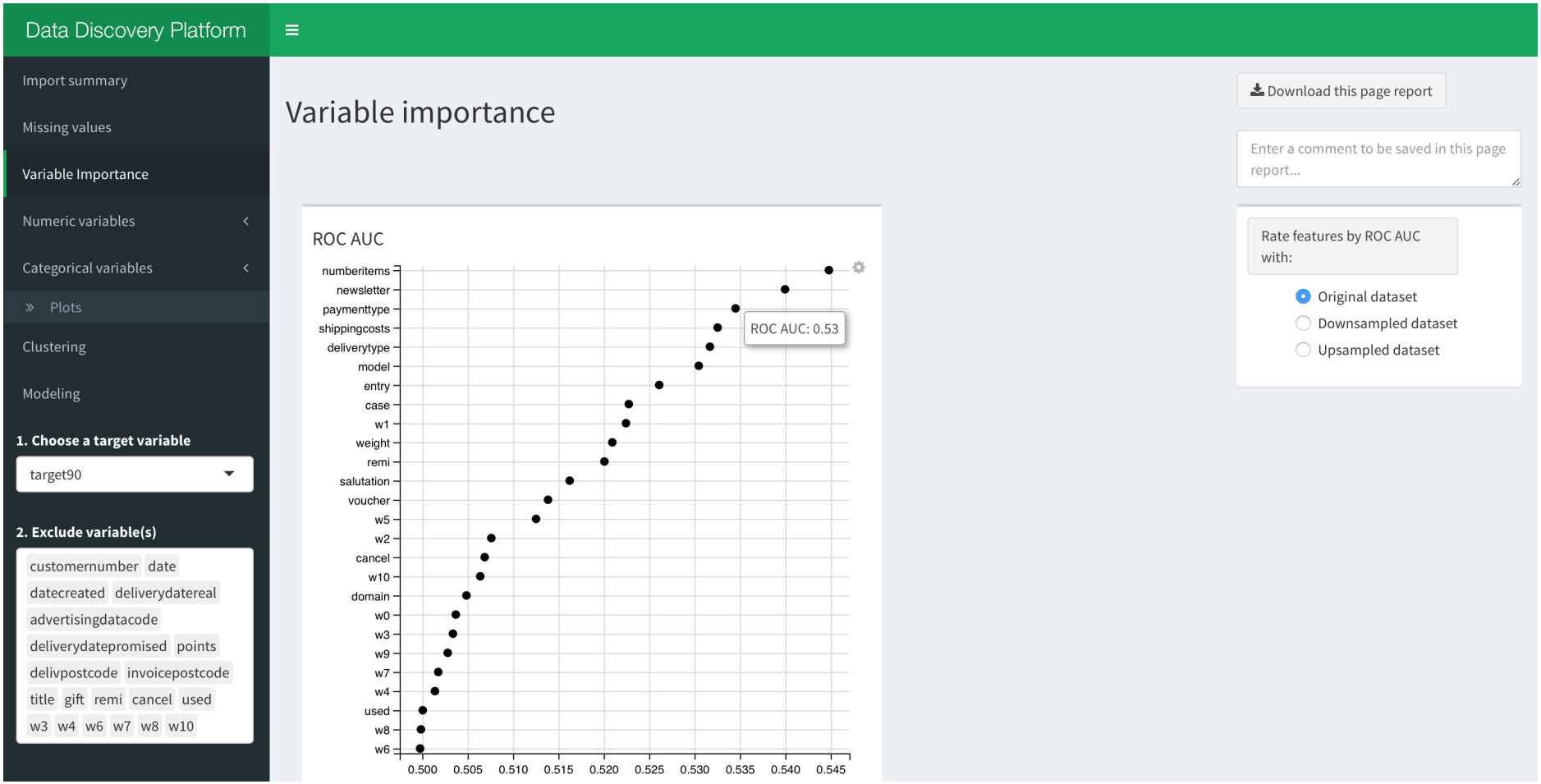


Figure 21: *Variable importance* tab

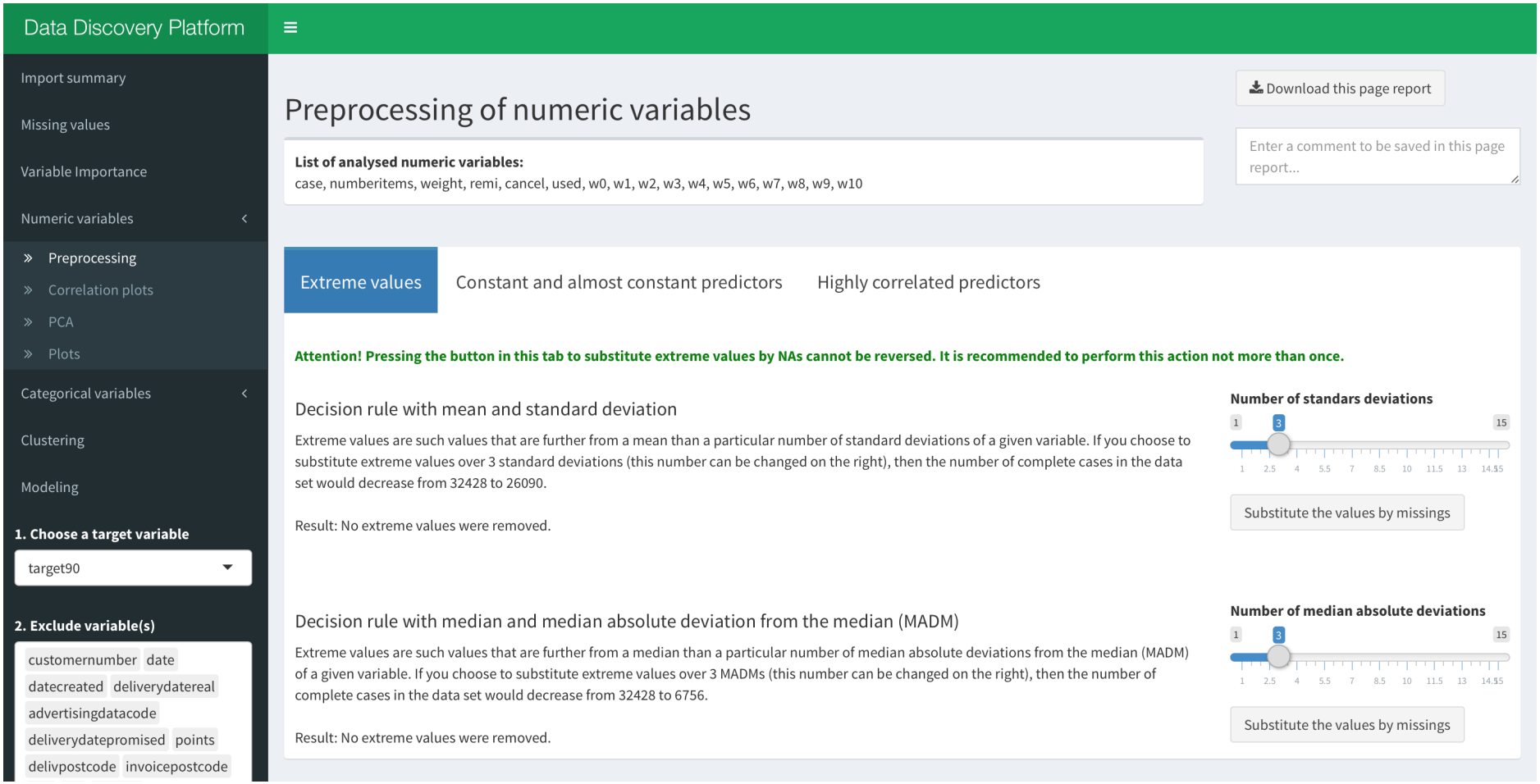
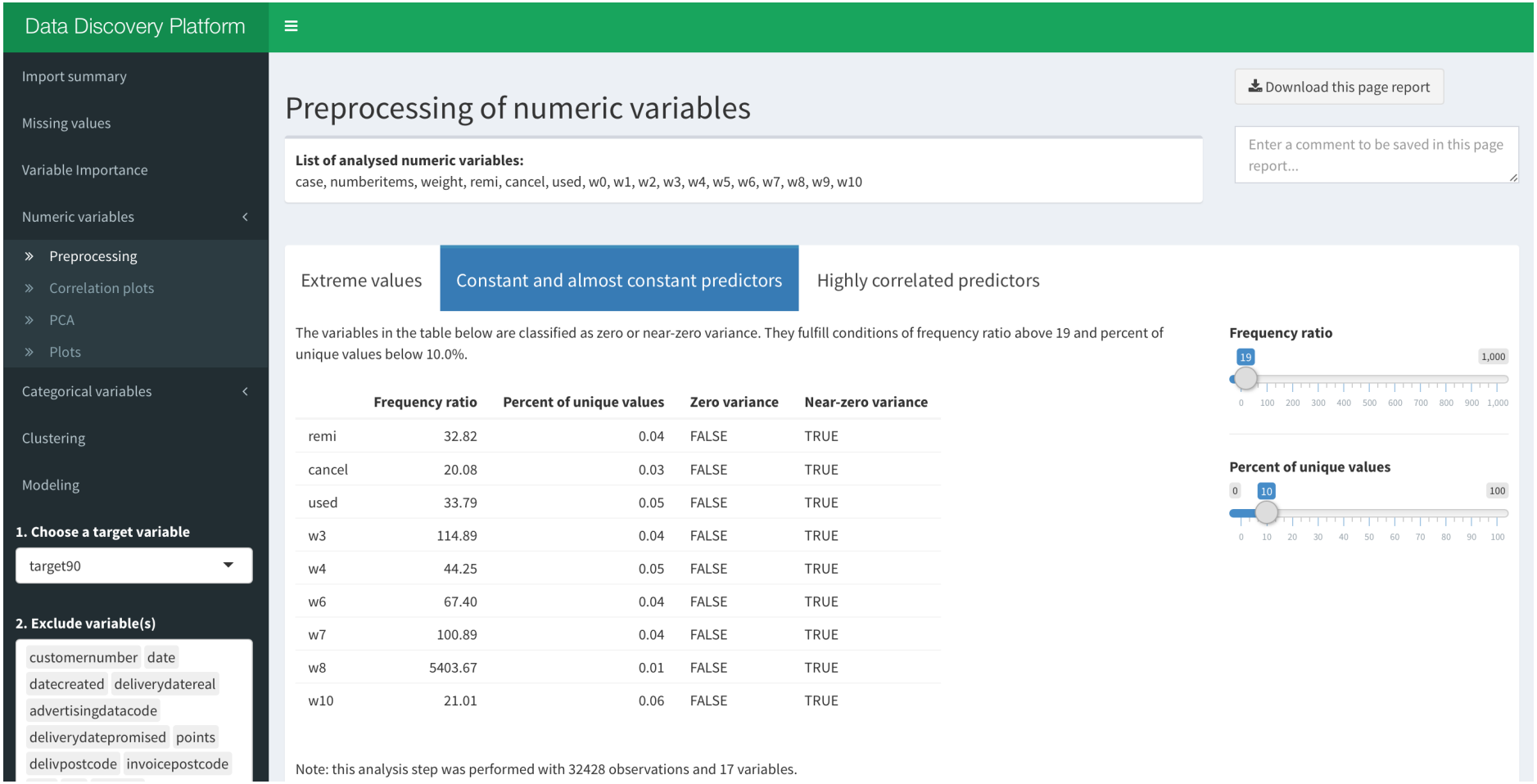


Figure 22: *Numeric variable* > *Preprocessing* > *Extreme values* tab



	Frequency ratio	Percent of unique values	Zero variance	Near-zero variance
remi	32.82	0.04	FALSE	TRUE
cancel	20.08	0.03	FALSE	TRUE
used	33.79	0.05	FALSE	TRUE
w3	114.89	0.04	FALSE	TRUE
w4	44.25	0.05	FALSE	TRUE
w6	67.40	0.04	FALSE	TRUE
w7	100.89	0.04	FALSE	TRUE
w8	5403.67	0.01	FALSE	TRUE
w10	21.01	0.06	FALSE	TRUE

Figure 23: *Numeric variable > Preprocessing > Constant and almost constant predictors* tab

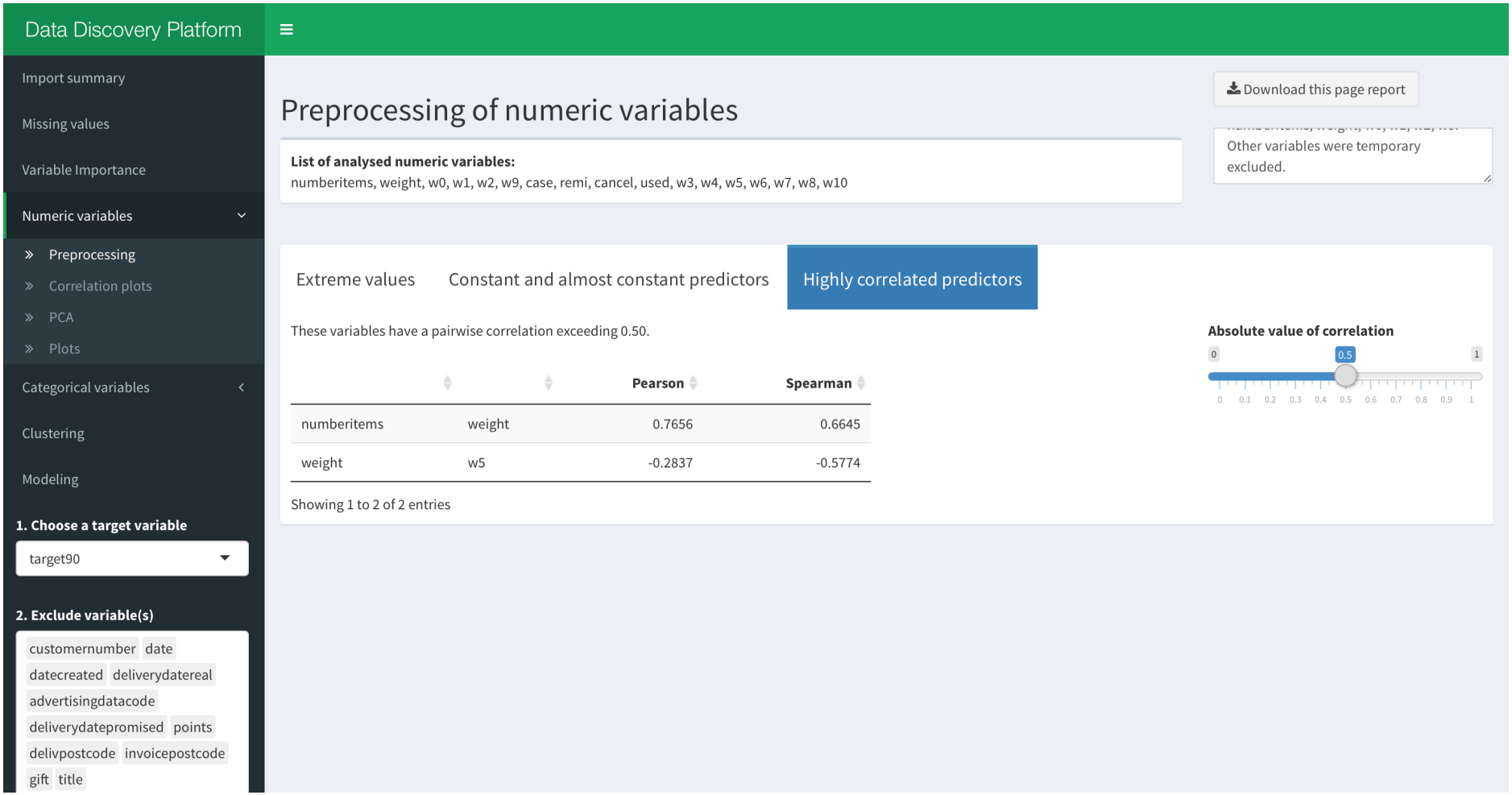


Figure 24: *Numeric variable* > *Preprocessing* > *Highly correlated predictors* tab

Figure 25: *Numeric variable* > *Correlation plots* tab

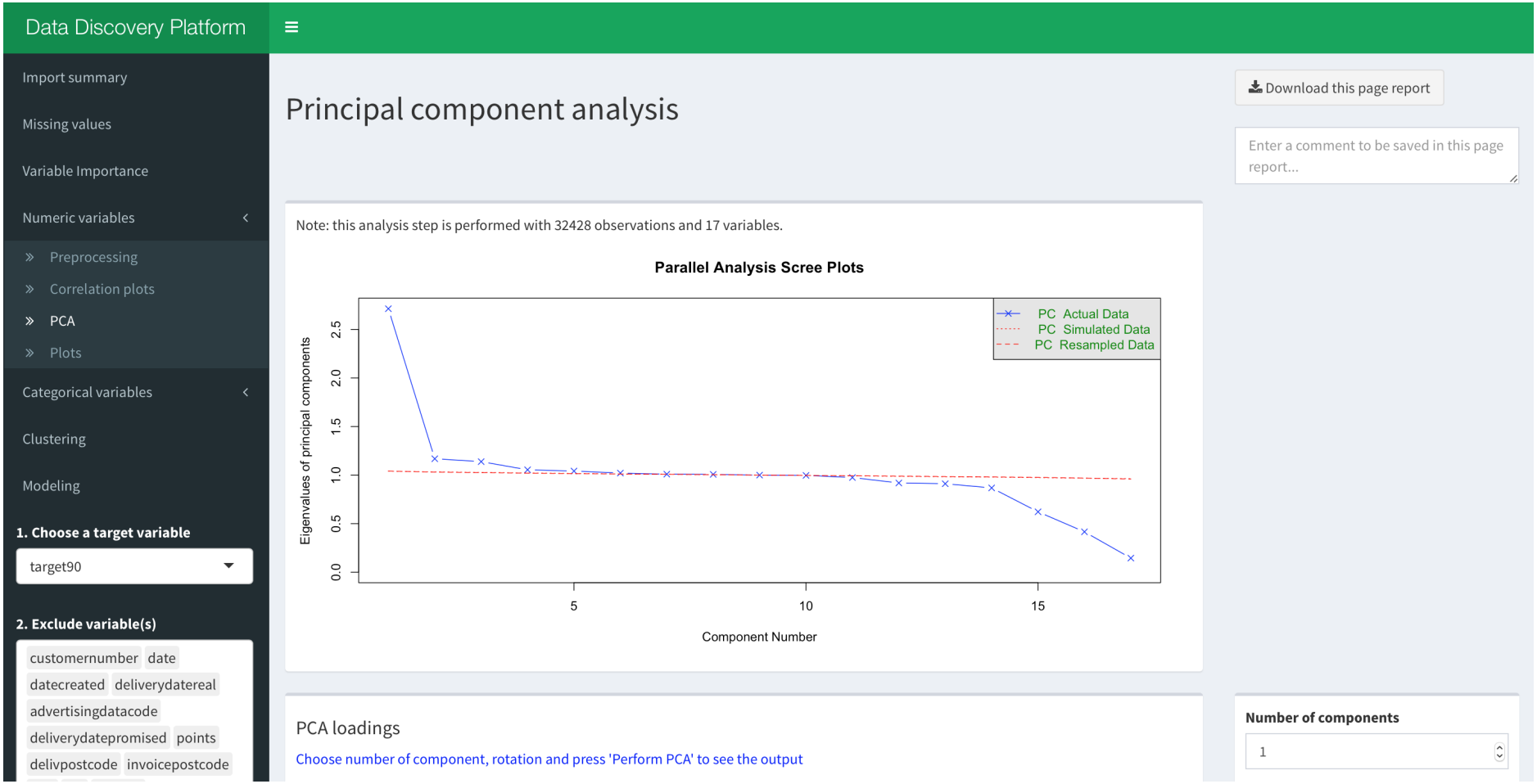


Figure 26: *Numeric variable* > *PCA* tab

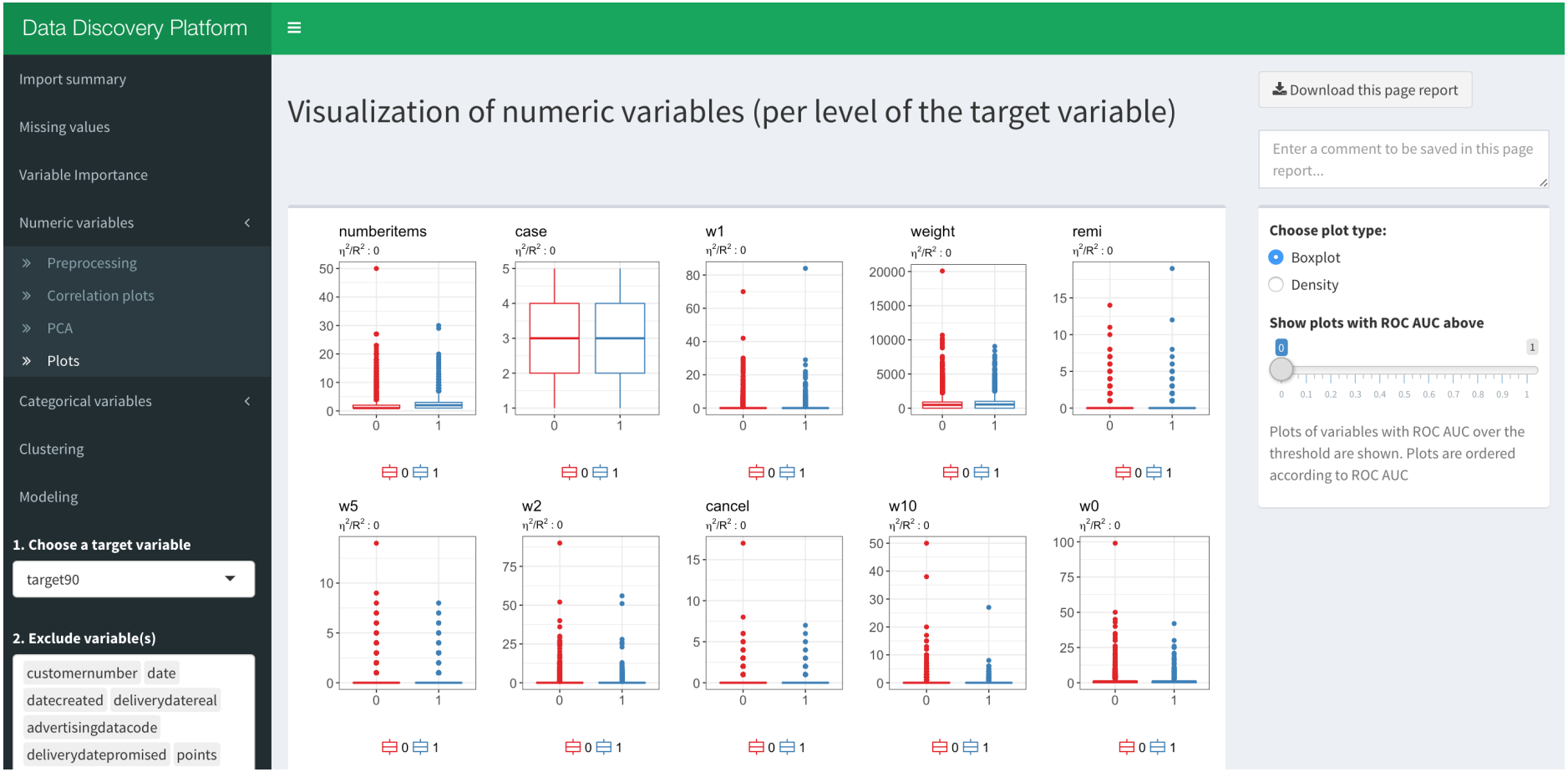


Figure 27: *Numeric variable > Plots* tab

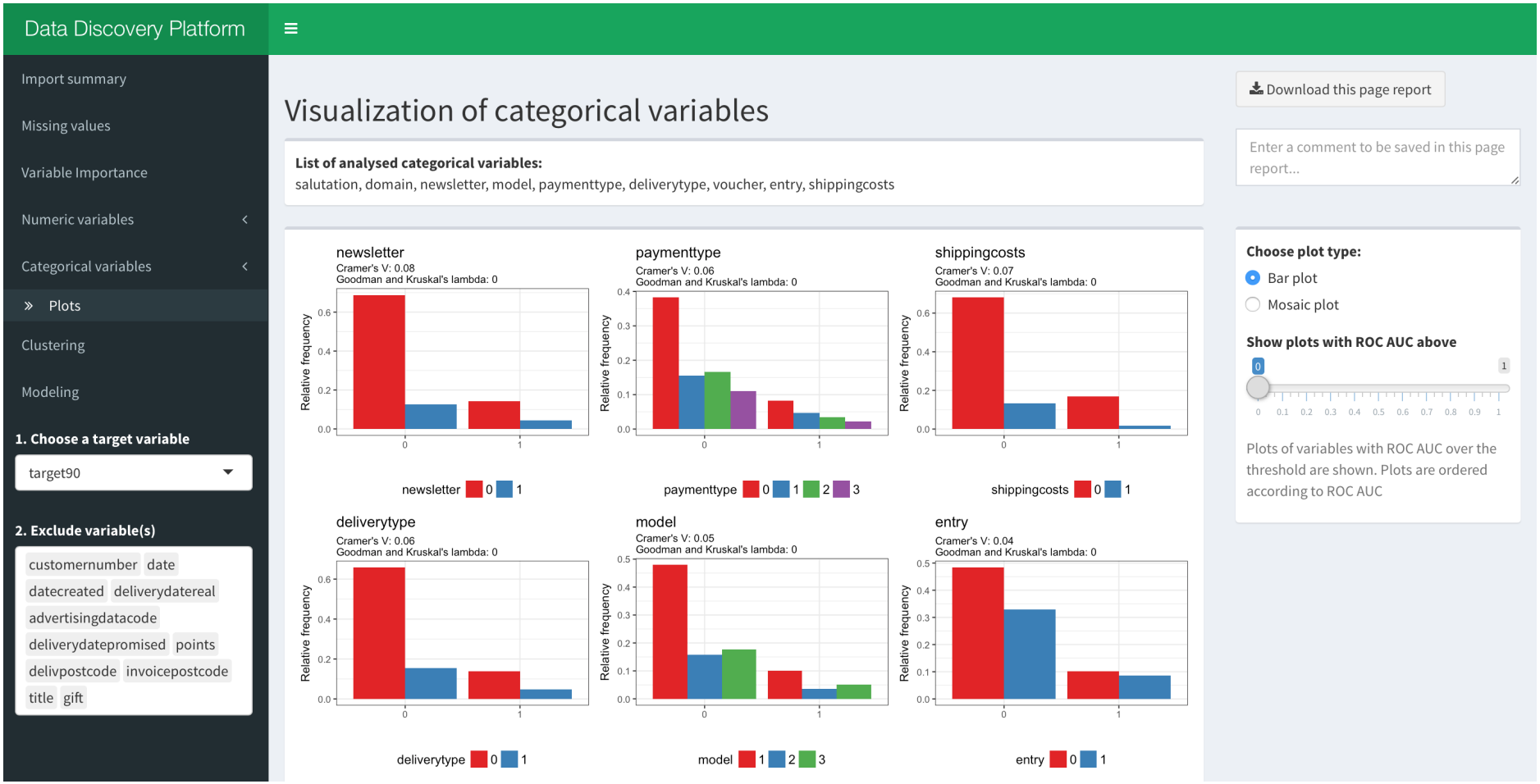


Figure 28: *Categorical variable > Plots* tab

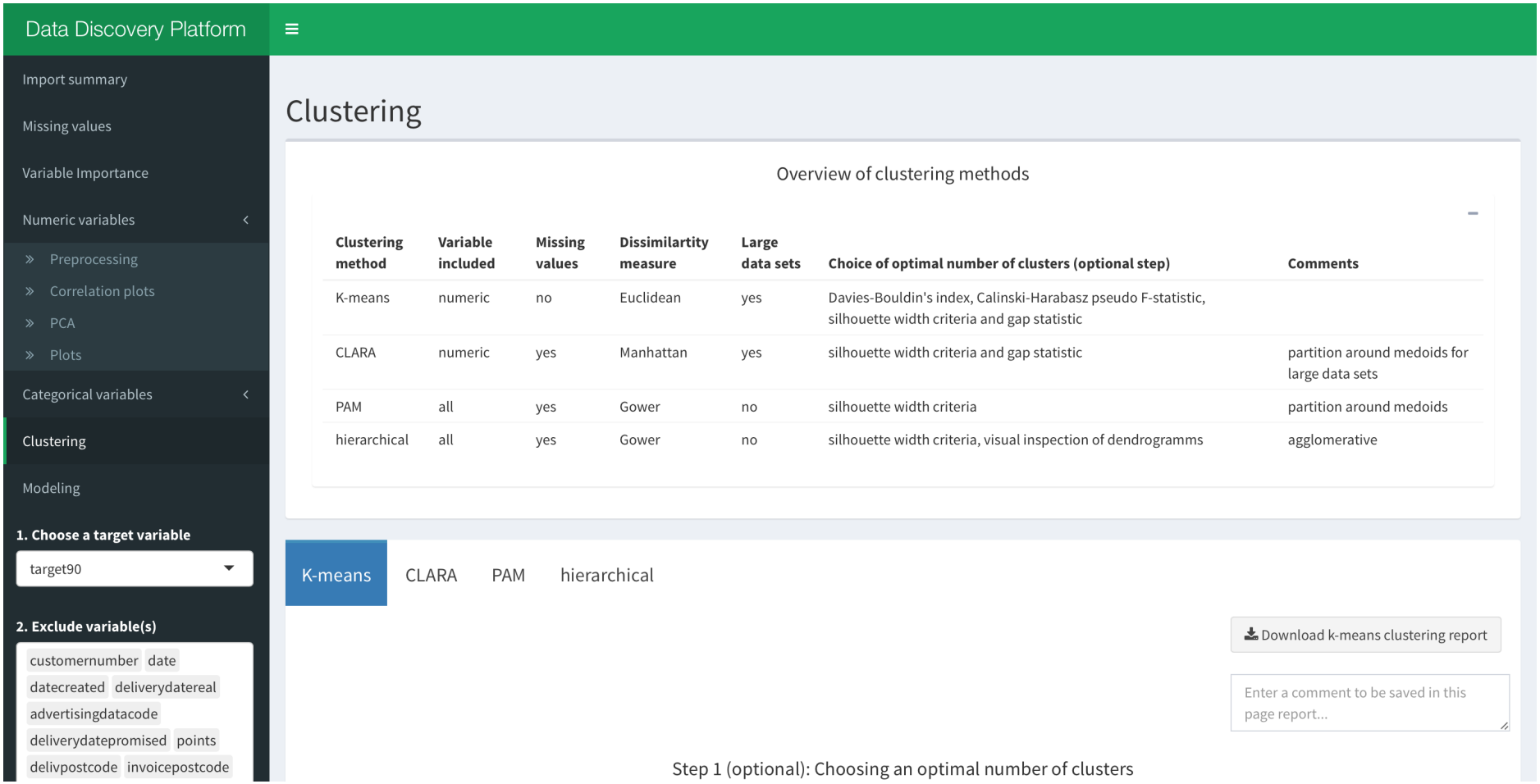


Figure 29: General view of the *Clustering* tab

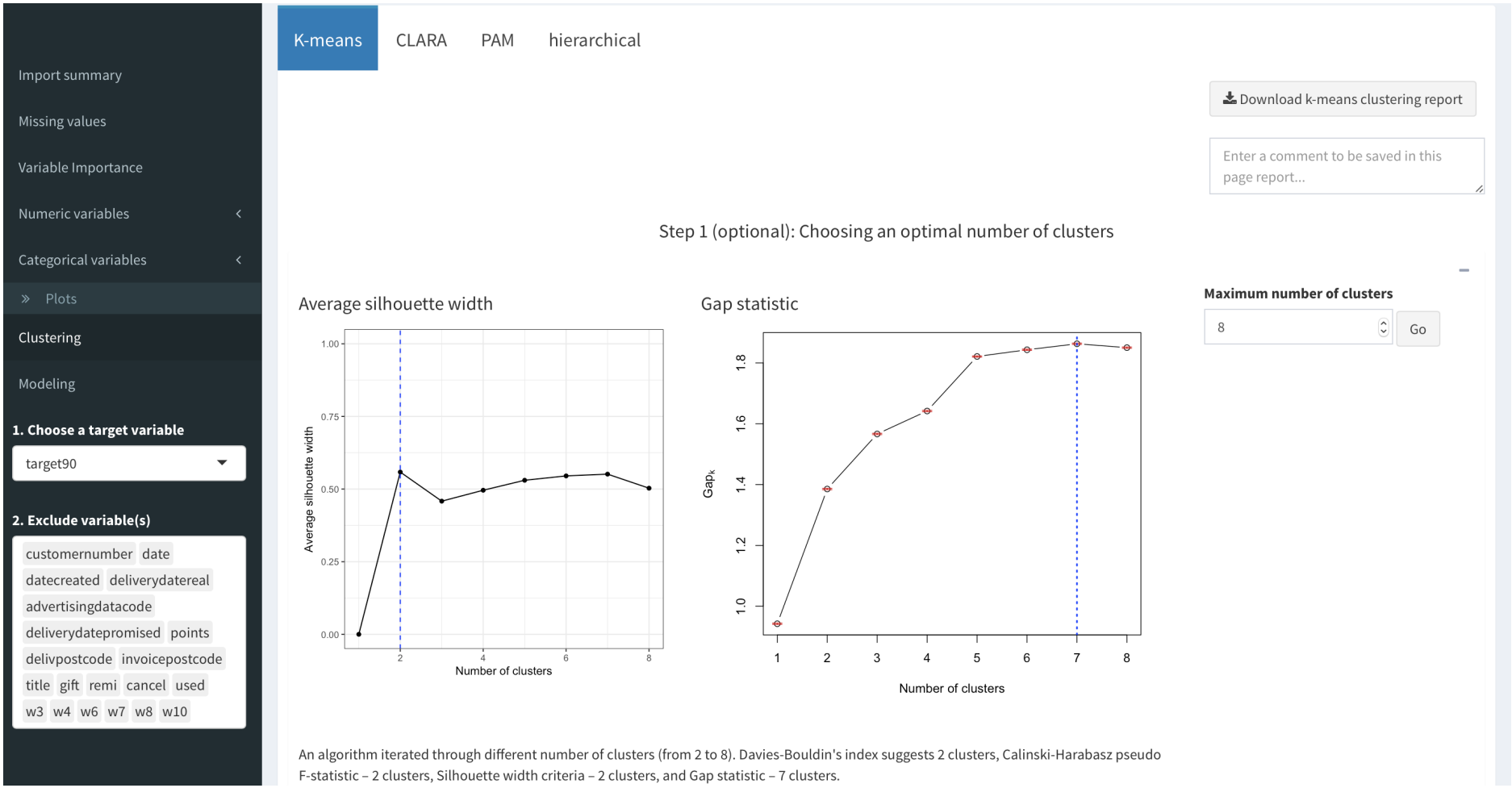


Figure 30: Step 1 of K-means in the *Clustering* tab

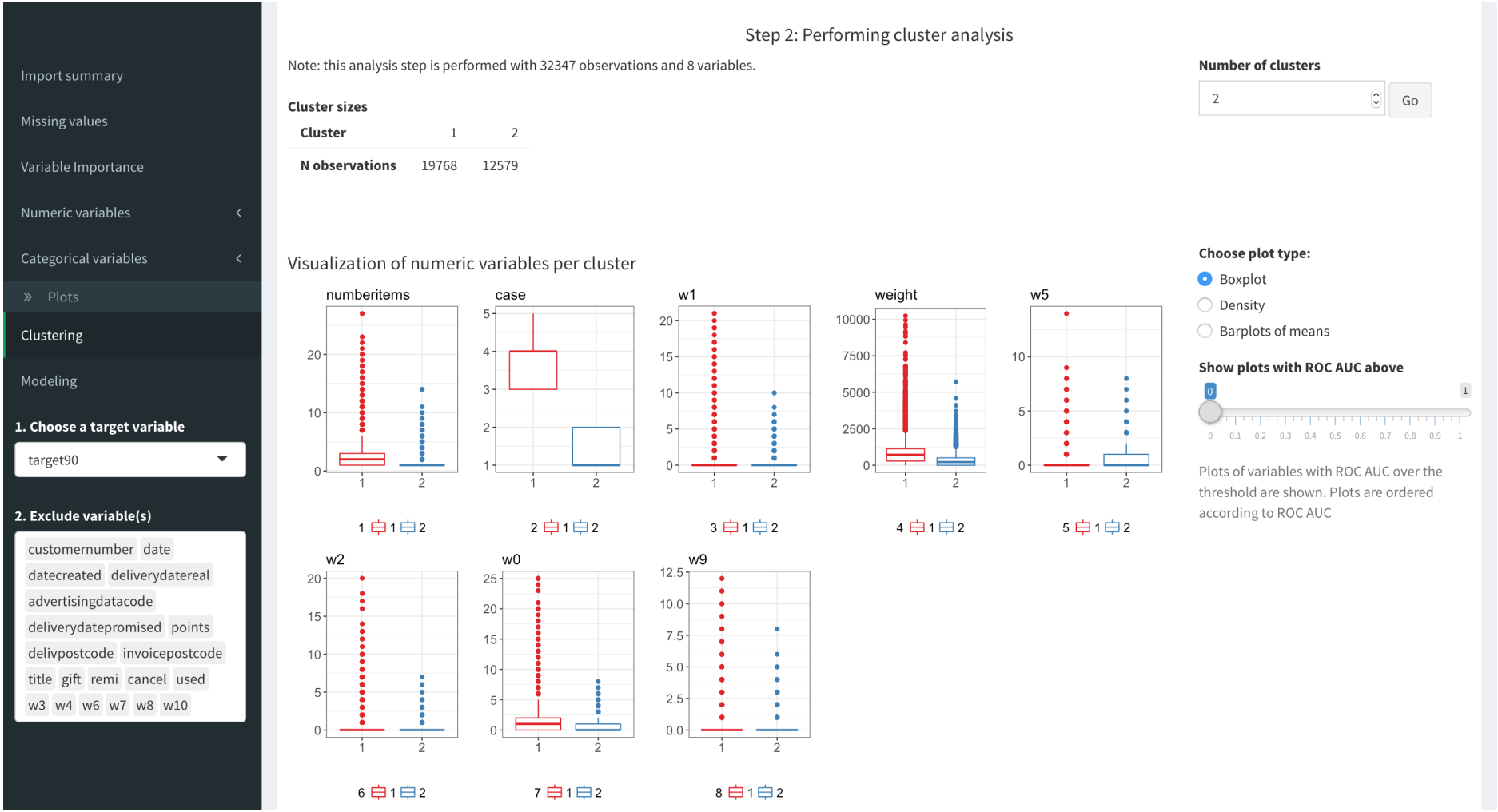


Figure 31: Step 2 of K-means in the *Clustering* tab

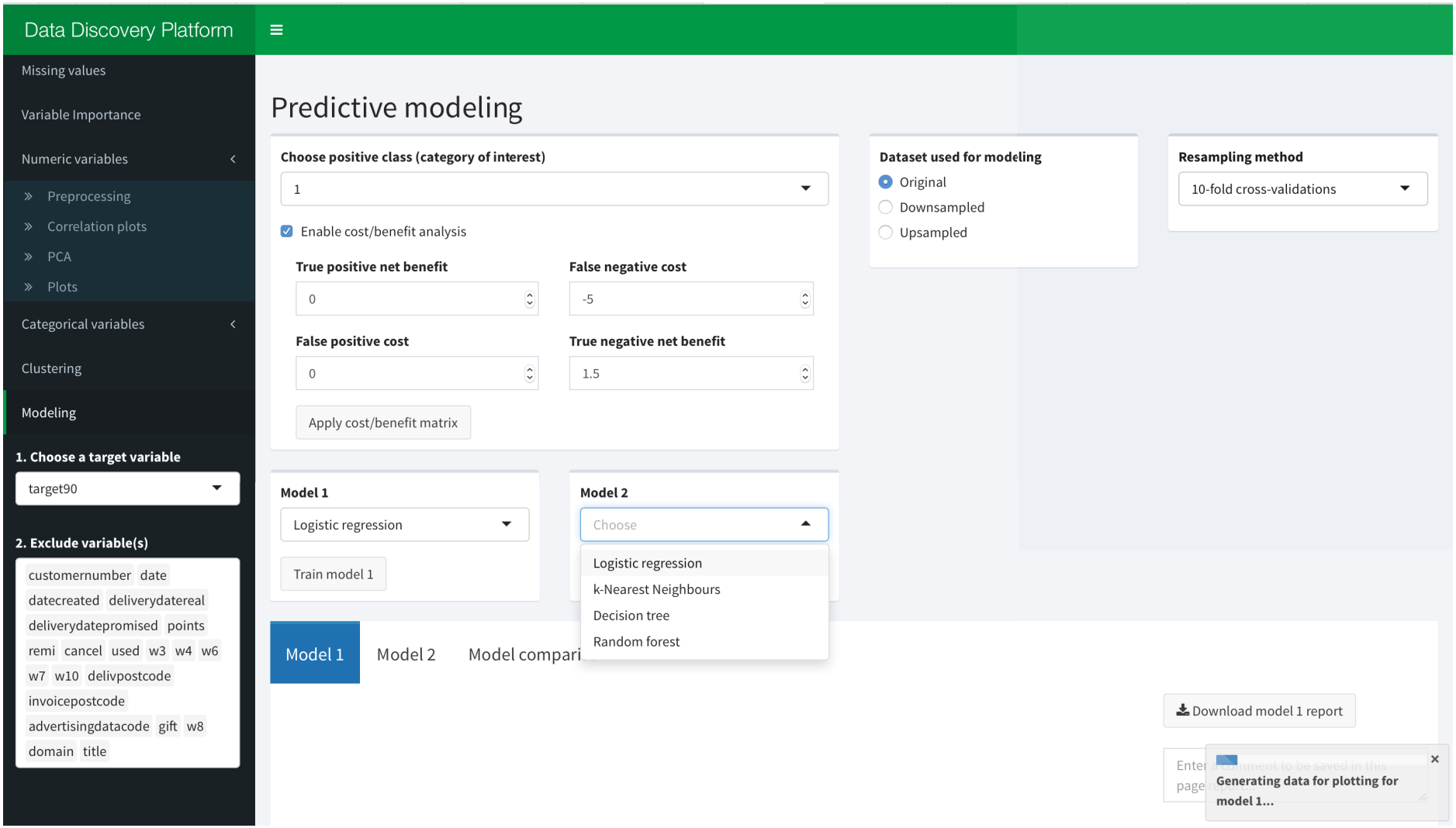


Figure 32: General view of the *Modeling* tab with enabled cost/benefit analysis

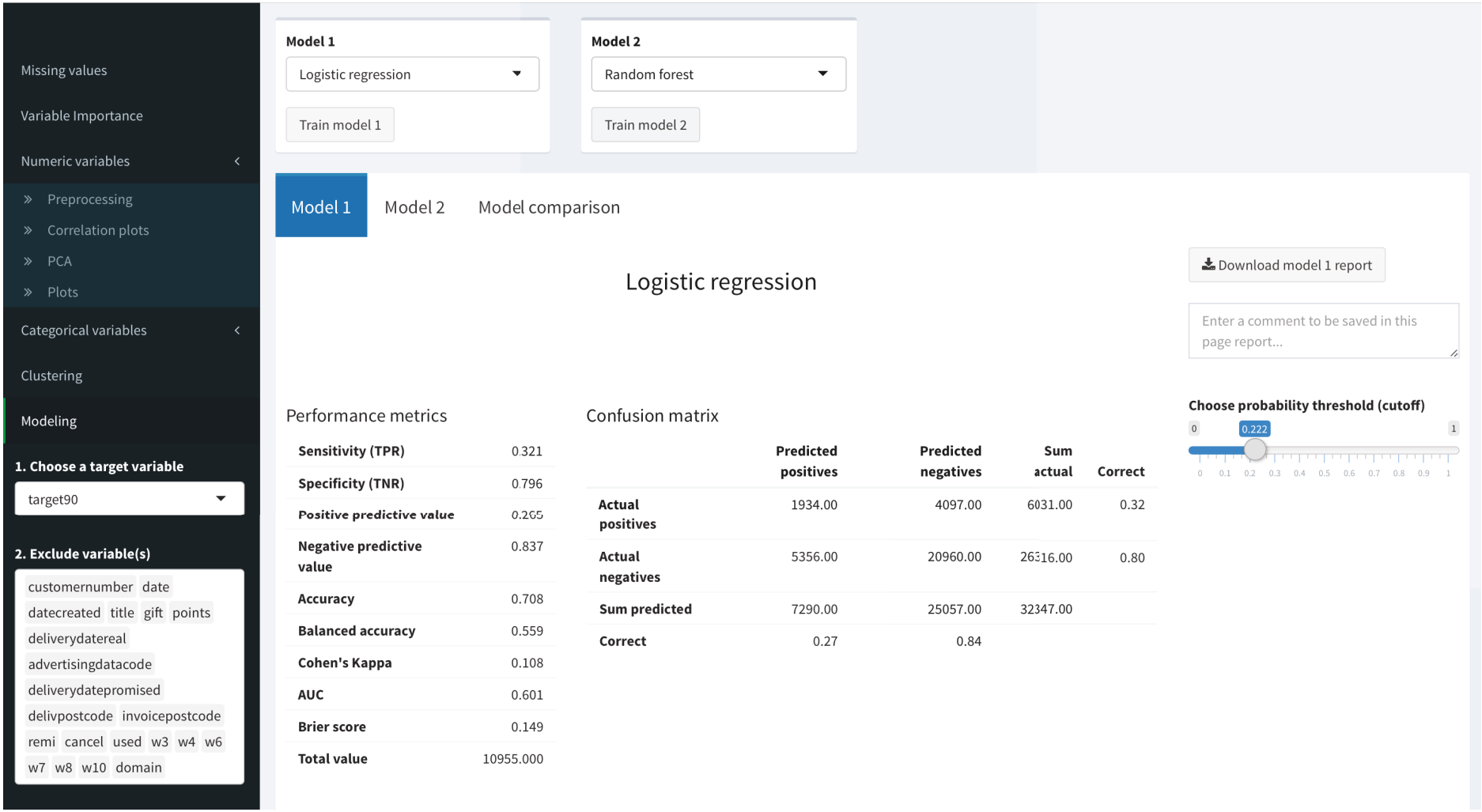


Figure 33: Numeric output of a classifier in the *Modeling* > *Model 1* tab

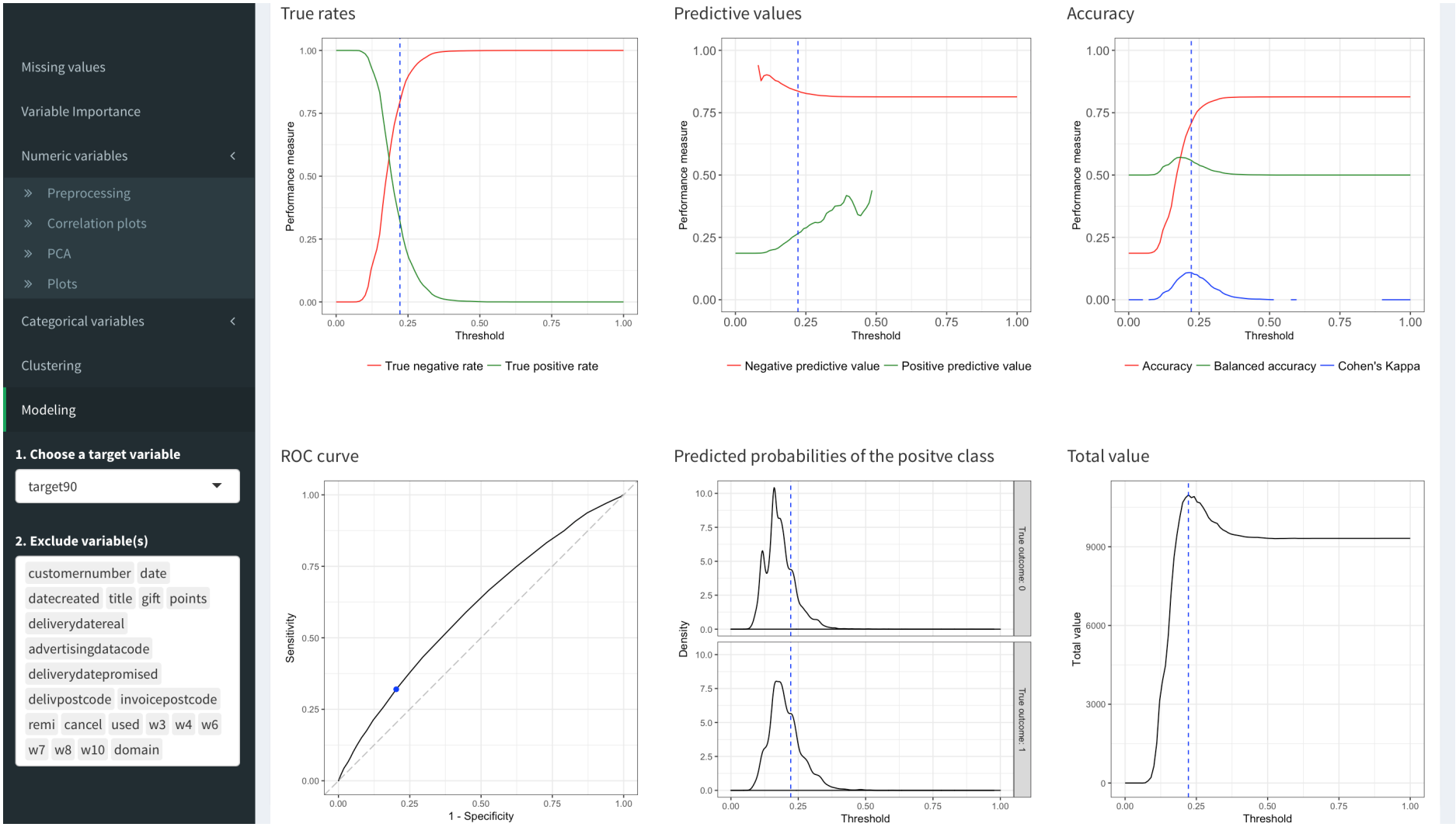


Figure 34: Graphical output of a classifier in the *Modeling* > *Model 1* tab

B List of files on the enclosed CD

Software outputs

Extended dataset (with PCA and clustering results)

dmc2010_train.csv-data-2017-02-19.csv

Metadata

dmc2010_train.csv-metadata-2017-02-19.csv

Report of the *Import summary* tab

dmc2010_train.csv-01-import-summary-2017-02-19.html

Report of the *Missing values* tab (matrix plot sorted by `deliverydatereal`)

dmc2010_train.csv-02-missing-values-2017-02-19_1.html

Report of the *Missing values* tab (matrix plot sorted by `paymenttype`)

dmc2010_train.csv-02-missing-values-2017-02-19_2.html

Report of the *Variable importance* tab

dmc2010_train.csv-03-variable-importance-2017-02-19.html

Report of the *Numeric variables* > *Preprocessing* tab

dmc2010_train.csv-04-preprocessing-2017-02-19.html

Report of the *Numeric variables* > *Correlation plots* tab

dmc2010_train.csv-05-correlation-plots-2017-02-19.html

Report of the *Numeric variables* > *PCA* tab

dmc2010_train.csv-06-PCA-2017-02-19.html

Report of the *Numeric variables* > *Plots* tab

dmc2010_train.csv-07-numeric-plots-2017-02-19.html

Report of the *Categorical variables* > *Plots* tab

dmc2010_train.csv-08-catergorical-plots-2017-02-19.html

Report of the *Clustering* > *K-means* tab (two-cluster solution)

dmc2010_train.csv-09-kmeans-clustering-2017-02-19_2.html

Report of the *Clustering* > *K-means* tab (five-cluster solution)

dmc2010_train.csv-09-kmeans-clustering-2017-02-19_5.html

Report of the *Clustering* > *CLARA* tab

dmc2010_train.csv-10-clara-clustering-2017-02-19.html

Report of the *Clustering* > *PAM* tab

dmc2010_train.csv-11-pam-clustering-2017-02-19.html

Report of the *Clustering* > *Hierarchical* tab

dmc2010_train.csv-12-hierarchical-clustering-2017-02-19.html

Report of the *Modeling* > *Model 1* tab (Logistic regression)

dmc2010_train.csv-13-model1-2017-02-19.html

Report of the *Modeling* > *Model 2* tab (Random forest)

dmc2010_train.csv-14-model2-2017-02-19.html

Report of the *Modeling* > *Model comparison* tab

dmc2010_train.csv-15-model-comparison-2017-02-19.html

In-app documentation

DDP-documentation-2017-02-19.html

Other files

Dataset by Data Mining Cup (2010)

dmc2010_train.csv

Competition task by Data Mining Cup (2010)

dmc2010_task.pdf

Features description by Data Mining Cup (2010)

dmc2010_features.pdf

References

Agresti, A. (2002). *Categorical Data Analysis*. Wiley-Interscience, New York, 2nd edition.

Allaire, J., Cheng, J., Xie, Y., McPherson, J., Chang, W., Allen, J., Wickham, H., Atkins, A., and Hyndman, R. (2016). *rmarkdown: Dynamic Documents for R*. R package version 1.3. URL https://CRAN.R-project.org/package=rmarkdown.

Auguie, B. (2016). *gridExtra: Miscellaneous Functions for "Grid" Graphics*. R package version 2.2.1. URL https://CRAN.R-project.org/package=gridExtra.

Bailey, E. (2015). *shinyBS: Twitter Bootstrap Components for Shiny*. R package version 0.61. URL https://CRAN.R-project.org/package=shinyBS.

Barnett, V. and Lewis, T. (1994). *Outliers in Statistical Data*. Wiley, Chichester; New York, 3rd edition.

Batista, G., Prati, R. C., and Monard, M. C. (2004). A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29.

Behrens, J. T. (1997). Principles and Procedures of Exploratory Data Analysis. *Psychological Methods*, 2(2):131–160.

Bischl, B., Lang, M., Richter, J., Bossek, J., Judt, L., Kuehn, T., Studerus, E., Kotthoff, L., and Julia, S. (2016). *mlr: Machine Learning in R*. R package version 2.8. URL https://CRAN.R-project.org/package=mlr.

Bortz, J. and Lienert, G. A. (2008). *Kurzgefasste Statistik für die klinische Forschung Leitfaden für die verteilungsfreie Analyse kleiner Stichproben; mit 97 Tabellen*. Springer-Lehrbuch: Bachelor, Master. Springer, Heidelberg, 3rd edition.

Bortz, J. and Schuster, C. (2010). *Statistik für Human- und Sozialwissenschaftler mit 163 Tabellen*. Springer-Lehrbuch. Springer, Berlin Heidelberg, 7th edition.

Breiman, L., Cutler, A., Liaw, A., and Wiener, M. (2015). *randomForest: Breiman and Cutler's Random Forests for Classification and Regression*. R package version 4.6-12. URL https://CRAN.R-project.org/package=randomForest.

Chang, W. (2016). *shinydashboard: Create Dashboards with "Shiny"*. R package version 0.5.3. URL https://CRAN.R-project.org/package=shinydashboard.

Chang, W., Cheng, J., Allaire, J., Xie, Y., and McPherson, J. (2016). *shiny: Web Application Framework for R*. R package version 0.14.2. URL https://CRAN.R-project.org/package=shiny.

Chang, W. and Wickham, H. (2016). *ggvis: Interactive Grammar of Graphics*. R package version 0.4.3. URL https://CRAN.R-project.org/package=ggvis.

Cheng, J. and Galili, T. (2016). *d3heatmap: Interactive Heat Maps Using "htmlwidgets" and "D3.js"*. R package version 0.6.1.1. URL https://CRAN.R-project.org/package=d3heatmap.

Clarke, B. S., Fokoué, E., and Zhang, H. H. (2009). *Principles and Theory for Data mining and Machine Learning*. Springer, Dordrecht; New York.

Cleveland, W. S. (1984). Graphical Methods for Data Presentation – Full Scale Breaks, Dot Charts, and Multibased Logging. *American Statistician*, 38(4):270–280.

Cleveland, W. S. and McGill, R. (1984). Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods. *Journal of the American Statistical Association*, 79(387):531–554.

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46.

Cooper, A., Reimann, R., and Cronin, D. (2007). *About Face 3: The Essentials of Interaction Design*. Wiley, 3rd edition.

Crawley, M. J. (2013). *The R book*. Wiley, Chichester, West Sussex, United Kingdom, 2nd edition.

Daróczi, G. and Tsegelskyi, R. (2015). *pander: An R Pandoc Writer*. R package version 0.6.0. URL https://CRAN.R-project.org/package=pander.

Data Mining Cup (2010). DMC 2010 Task. URL http://www.data-mining-cup.de/en/review/goto/article/dmc-2010.html. Last accessed on Jan 15, 2017.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. John Wiley and Sons, Inc., 2nd edition.

Enders, C. K. (2010). *Applied Missing Data Analysis*. Guilford Press.

Everitt, B. (1992). *The Analysis of Contingency Tables*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 2nd edition.

Everitt, B. S., Landau, S., and Leese, M. (2011). *Cluster Analysis*. Wiley, Chichester, West Sussex, U.K., 5th edition.

Ferri, C., Hernández-Orallo, J., and Modroiu, R. (2009). An Experimental Comparison of Performance Measures for Classification. *Pattern Recognition Letters*, 30(1):27 – 38.

Flach, P. A. (2012). *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, Cambridge; New York.

Goodwin, K. (2009). *Designing for the Digital Age: How to Create Human-Centered Products and Services*. Wiley, Indianapolis, IN.

Hand, D. J. (2009). Measuring Classifier Performance: a Coherent Alternative to the Area Under the ROC Curve. *Machine Learning*, 77(1):103–123.

Hand, D. J., Mannila, H., and Smyth, P. (2001). *Principles of Data Mining*. MIT Press, Cambridge, Mass.

Hand, D. J. and Till, R. J. (2001). A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. *Machine Learning*, 45(2):171–186.

Härdle, W., Lee, Y.-J., Schäfer, D., and Yeh, Y.-R. (2009). Variable Selection and Oversampling in the Use of Smooth Support Vector Machines for Predicting the Default Risk of Companies. *Journal of Forecasting*, 28(6):512–534.

Härdle, W. K., Klinke, S., and Rönz, B. (2015). *Introduction to Statistics: Using Interactive MM*Stat Elements*. Springer International Publishing.

Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, NY, 2nd edition.

Hechenbichler, K. and Schliep, K. (2004). Weighted k-Nearest-Neighbor Techniques and Ordinal Classification. URL http://nbn-resolving.de/urn/resolver.pl?urn=nbn:de:bvb:19-epub-1769-9. Last accessed on Nov 20, 2016.

Hilbe, J. (2009). *Logistic Regression Models*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis.

Hoaglin, D. C., Mosteller, F., and Tukey, J. W. (1983). *Understanding Robust and Exploratory Data Analysis*. Wiley, New York.

Hosmer, D. W., Lemeshow, S., and Sturdivant, R. X. (2013). *Applied logistic regression*. Wiley series in probability and statistics. Wiley, Hoboken, New Jersey, 3rd edition.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer, New York.

Japkowicz, N. and Shah, M. (2011). *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press.

Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer, New York, 2nd edition.

Kaufman, L. and Rousseeuw, P. J. (1990). *Finding Groups in Data: an Introduction to Cluster Analysis*. Wiley, New York.

Kohavi, R. (1995). A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence – Volume 2*, pages 1137–1143. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Kotz, S., Balakrishnan, N., Read, C. B., and Vidakovic, B. (2006). *Encyclopedia of Statistical Sciences*. Wiley-Interscience, Hoboken, N.J., 2nd edition.

Krug, S. (2010). *Web Usability: Rocket Surgery Made Easy*. Addison-Wesley, München.

Krug, S. (2014). *Don't Make Me Think: Web and Mobile Usability – Das intuitive Web*. mipt Verlag, 3rd edition.

Kuhn, M. (2008). Building Predictive Models in R Using the caret Package. *Journal of Statistical Software*, 28(5):1–26.

Kuhn, M. (2016). *caret: Classification and Regression Training*. R package version 6.0-73. URL https://CRAN.R-project.org/package=caret.

Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modeling*. Springer, New York.

Kuniavsky, M. (2003). *Observing the User Experience: A Practitioner's Guide to User Research*. Morgan Kaufmann Publishers, San Francisco, CA.

Larose, D. T. and Larose, C. D. (2015). *Data Mining and Predictive Analytics*. John Wiley and Sons Inc., Hoboken, New Jersey, 2nd edition.

Little, R. J. A. and Rubin, D. B. (2002). *Statistical Analysis with Missing Data*. Chichester: Wiley, 2nd edition.

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., and Hornik, K. (2016). *cluster: Cluster Analysis Basics and Extensions*. R package version 2.0.5. URL https://CRAN.R-project.org/package=cluster.

Meyer, D., Zeileis, A., and Hornik, K. (2016). *vcd: Visualizing Categorical Data*. R package version 1.4-3. URL https://CRAN.R-project.org/package=vcd.

Milligan, G. W. and Cooper, M. C. (1988). A Study of Standardization of Variables in Cluster Analysis. *Journal of Classification*, 5(2):181–204.

Müllner, D. (2013). fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python. *Journal of Statistical Software*, 53(9):1–18.

Myatt, G. J. and Johnson, W. P. (2009). *Making sense of Data II: A Practical Guide to Data Visualization, Advanced Data Mining Methods, and Applications*. John Wiley and Sons, Hoboken, N.J.

Myatt, G. J. and Johnson, W. P. (2014). *Making Sense of Data I: A Practical Guide to Exploratory Data Analysis and Data Mining*. John Wiley and Sons Inc., Hoboken, New Jersey, 2nd edition.

Neuwirth, E. (2014). *RColorBrewer: ColorBrewer Palettes*. R package version 1.1-2. URL https://CRAN.R-project.org/package=RColorBrewer.

Pearson, R. K. (2011). *Exploring Data in Engineering, the Sciences, and Medicine*. Oxford University Press, Oxford; New York.

Pedersen, T. L. (2016). *shinyFiles: A Server-Side File System Viewer for Shiny*. R package version 0.6.2. URL https://CRAN.R-project.org/package=shinyFiles.

Piegorsch, W. W. (2015). *Statistical Data Analytics: Foundations for Data Mining, Informatics, and Knowledge Discovery*. John Wiley and Sons Inc., Chichester, West Sussex.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org.

Revelle, W. (2016). *psych: Procedures for Psychological, Psychometric, and Personality Research*. Northwestern University, Evanston, Illinois. R package version 1.6.9. URL https://CRAN.R-project.org/package=psych.

Ripley, B. and Venables, W. (2016). *nnet: Feed-Forward Neural Networks and Multinomial Log-Linear Models*. R package version 7.3-12. URL https://CRAN.R-project.org/package=nnet.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge.

Rousseeuw, P. J. (1987). Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65.

Saffer, D. (2007). *Designing for Interaction: Creating Smart Applications and Clever Devices*. New Riders: Published in association with AIGA Design Press, Berkeley CA.

Schliep, K. and Hechenbichler, K. (2016). *kknn: Weighted k-Nearest Neighbors*. R package version 1.3.1. URL https://CRAN.R-project.org/package=kknn.

Schwartz, M. (2015). R code for miscellaneous measures of association. URL https://gist.github.com/marcschwartz/3665743. Last accessed on Dec 10, 2016.

Shmueli, G. (2010). To explain or to Predict? *Statistical science*, 25(3):289–310.

Sprent, P. (1998). *Data Driven Statistical Methods*. Chapman and Hall texts in statistical science series. Chapman and Hall, London; New York.

Steinley, D. (2004). Standardizing variables in k-means clustering. In Banks, D., McMorris, F. R., Arabie, P., and Gaul, W., editors, *Classification, Clustering, and Data Mining Applications: Proceedings of the Meeting of the International Federation of Classification Societies (IFCS), Illinois Institute of Technology, Chicago, 15–18 July 2004*, pages 53–60. Springer Berlin Heidelberg, Berlin, Heidelberg.

Tabachnick, B. G. and Fidell, L. S. (2001). *Using Multivariate Statistics*. Allyn & Bacon, Inc., Needham Heights, MA, USA, 4th edition.

Templ, M., Alfons, A., and Filzmoser, P. (2012). Exploring Incomplete Data using Visualization Techniques. *Advances in Data Analysis and Classification*, 6(1):29–47.

Templ, M., Alfons, A., Kowarik, A., and Prantner, B. (2016). *VIM: Visualization and Imputation of Missing Values*. R package version 4.6.0. URL https://cran.r-project.org/package=VIM.

Therneau, T., Atkinson, B., and Ripley, B. (2015). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-10. URL https://CRAN.R-project.org/package=rpart.

Tibshirani, R., Walther, G., and Hastie, T. (2001). Estimating the Number of Clusters in a Dataset via the Gap Statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423.

Tidwell, J. (2011). *Designing Interfaces*. O'Reilly, Sebastopol, CA, 2nd edition.

Tufte, E. R. (2001). *The Visual Display of Quantitative Information*. Graphics Press, 2nd edition.

Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley Pub. Co., Reading, Mass.

Unwin, A. (2015). *Graphical Data Analysis with R*. CRC Press.

Upton, G. and Cook, I. (2002). *A Dictionary of Statistics*. Oxford University Press.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, 4th edition.

Walesiak, M. and Dudek, A. (2016). *clusterSim: Searching for Optimal Clustering Procedure for a Data Set*. R package version 0.45-1. URL https://CRAN.R-project.org/package=clusterSim.

Wickham, H. (2007). Reshaping Data with the reshape Package. *Journal of Statistical Software*, 21(12):1–20.

Wickham, H. (2009). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Wickham, H. (2017). *forcats: Tools for Working with Categorical Variables (Factors)*. R package version 0.2.0. URL https://CRAN.R-project.org/package=forcats.

Wickham, H., Hester, J., and Francois, R. (2016). *readr: Read Tabular Data*. R package version 1.0.0. URL https://CRAN.R-project.org/package=readr.

Wilcox, R. R. (2010). *Fundamentals of Modern Statistical Methods: Substantially Improving Power and Accuracy*. Springer, New York, NY, 2nd edition.

Xie, Y. (2016). *DT: A Wrapper of the JavaScript Library "DataTables"*. R package version 0.2. URL https://CRAN.R-project.org/package=DT.

Yu, C. H. (2010). Exploratory Data Analysis in the Context of Data Mining and Resampling. *International Journal of Psychological Research*, 3(1):9–22.

Ehrenwörtlichen Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung von anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremdem Quellen direct oder indirekt übernommene Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder änlicher Form keiner anderen Prüfungsbehörder vorgelegt.

Berlin, den 11. März 2017